Universidade Federal Rural do Semi-Árido
Programa de Pós-Graduação em Sistemas de Comunicação e Automação

Jonathan Paulo Pinheiro Pereira

Aplicação de Algoritmos Genéticos ao problema de planejamento de caminhos com a abordagem de Decomposição em Células Convexas para o caso aproximado

MOSSORÓ-RN
2012

Universidade Federal Rural do Semi-Árido
Programa de Pós-Graduação em Sistemas de Comunicação e Automação

Jonathan Paulo Pinheiro Pereira

Aplicação de Algoritmos Genéticos ao problema de planejamento de caminhos com a abordagem de Decomposição em Células Convexas para o caso aproximado

Orientador: Prof. Dr. José Patrocínio da Silva

Co-orientador: Prof. Dr. Idalmir de Souza Queiroz Junior

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Sistemas de Comunicação e Automação da UFERSA (área de concentração: Controle e Sistemas de Energia) como parte dos requisitos para obtenção do título de Mestre em Ciências.

MOSSORÓ-RN
2012

Ficha catalográfica preparada pelo setor de classificação e catalogação da Biblioteca "Orlando Teixeira" da UFERSA

P436a Pereira, Jonathan Paulo Pinheiro.
Aplicação de algoritmos genéticos ao problema de planejamento de caminhos com a abordagem de decomposição em células convexas para o caso aproximado. / Jonathan Paulo Pinheiro Pereira. -- Mossoró, 2012.
77 f.: il.

Dissertação (Mestrado em Sistemas de Comunicação e Automação) - Universidade Federal Rural do Semi-Árido.
Orientador: Dr. José do Patrocínio da Silva
Co-orientador: Dr. Idalmir de Souza Queiroz Júnior

1. Algoritmos genéticos. 2. Planejamento de caminhos. 3. Decomposição em células. I. Título.

CDD: 006.31

Bibliotecária: Vanessa de Oliveira Pessoa
CRB15/453

Jonathan Paulo Pinheiro Pereira

Aplicação de Algoritmos Genéticos ao problema de planejamento de caminhos com a abordagem de Decomposição em Células Convexas para o caso aproximado

Dissertação de Mestrado aprovada em 26 de outubro de 2012 pela banca examinadora composta pelos seguintes membros:

Prof. Dr. José Patrocínio da Silva (Orientador) ..... UFERSA

Prof. Dr. Idalmir de Souza Queiroz Junior (Examinador Interno) ..... UFERSA

Prof. Dr. Antônio Ronaldo Gomes Garcia (Examinador Interno) ..... UFERSA

Profª Drª Heliana Bezerra Soares (Examinadora Externa) ..... UFRN

"...O segredo do sucesso é a constância do propósito..."
Benjamin Disraeli

Agradecimentos

A Deus em primeiro lugar pelo maravilhoso dom da vida.

A minha esposa, Priscila pela paciência e incentivo durante a realização deste trabalho.

Aos meus pais, Osiris e Penha pelo apoio durante esta jornada.

Ao meu orientador e ao meu co-orientador, professores Patrocínio e Idalmir, sou grato pela orientação e por acreditarem no meu trabalho.

Aos amigos, Valter e Pauliara pelas sugestões e companhia.

Aos demais colegas de pós-graduação, pelo incentivo.

À UFERSA, pelo apoio estrutural.

Resumo

Este trabalho apresenta uma aplicação de Algoritmos Genéticos (AG) para resolver o problema de planejamento de caminhos utilizando o método de decomposição em células convexas aproximado. O algoritmo é usado para determinar o menor caminho entre os pontos de origem e destino no grafo de conectividade gerado pelo método de decomposição em células. O objetivo principal deste estudo, é a avaliação do desempenho no uso do AG de critério heurístico adaptativo em relação ao uso do algoritmo com critério guloso, Djikstra. Simulações computacionais utilizando a linguagem C mostram testes para verificar influência de fatores no custo computacional como tamanho da população e o método de inicialização dos cromossomos. A observação gráfica do espaço de configuração e geração da trajetória é obtida através de uma aplicação em SCILAB. Para validação experimental do algoritmo foi desenvolvida uma plataforma robótica móvel de acionamento diferencial que recebe os pontos da trajetória através de um enlace de rádio.

Palavras-chave: Algoritmos Genéticos, Planejamento de Caminhos, Decomposição em Células.

Abstract

This work presents an application of Genetic Algorithms (GA) to solve the path planning problem using the approximate cells decomposition method. The algorithm is used to determine the shortest path between the points of origin and destination in the connectivity graph generated by the cell decomposition method. The main objective of this study is to evaluate the performance in using the GA with adaptive heuristic criterium regarding the use of the algorithm with greedy criterium, Djikstra. Computer simulations using the C language show influence of the computational cost as population size and chromosomes initialization method. The observation graphical of the configuration space and the trajectory generation is obtained through an application SCILAB. For experimental validation of the algorithm was developed a mobile robotic platform that receives track point through a radio link.

Keywords: Genetic Algorithms, Path planning, Cells Decomposition.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Introdução

A utilização de robôs móveis, que durante muitos anos esteve restrita a laboratórios de pesquisa para estudos bastante específicos como, por exemplo, as aplicações aeroespaciais, que aos poucos ganhou espaço em grandes indústrias, na realização de atividades que exigiam precisão, velocidade e repetição. O custo sempre foi um fator impeditivo, mas a constante evolução das áreas de eletrônica e informática determinou o desenvolvimento do setor industrial, reduziu custos de produção e disseminou muitos produtos e equipamentos, entre eles os próprios robôs que atualmente são utilizados em indústrias menores e em residências como robôs de serviço. Os robôs de serviço são robôs que realizam atividades que auxiliam seres humanos, através da realização de trabalhos cansativos, perigosos e/ou repetitivos, incluindo as tarefas domésticas (SCHRAFT, 1994).

Para realização destas atividades, os robôs necessitam ter a capacidade de processar informações coletadas do ambiente, utilizando algoritmos que otimizem recursos como memória e tempo de processamento e possam fornecer soluções eficientes para que estes possam deslocar-se no ambiente sem colidir com obstáculos e percorrendo o menor trajeto, visando menor consumo de energia. Esta atividade constitui-se na navegação do robô móvel.

A navegação é um problema fundamental nas aplicações de robótica móvel. Sua resolução permite a realização da tarefa mais básica para um robô que é o seu deslocamento. A busca por soluções são temas de muitas pesquisas (BORENSTEIN et al., 1997). Este estudo integra na verdade uma série de outros subproblemas que dentre estes, podemos destacar, o planejamento de caminhos e a geração de trajetórias. O planejamento de caminhos é um problema tão antigo quanto a própria robótica móvel (BRUCE; VELOSO, 2002), não existindo um procedimento único de resolvê-lo. Seu estudo permite a escolha de uma rota ponderada, por critérios que vão desde a diminuição do gasto energético, até o desvio eficiente de obstáculos presentes no ambiente, partindo de um ponto inicial para atingir um destino final (SWAMINATHAN, 2006).

A navegação, objeto deste estudo, é uma das principais atividades que o robô móvel deve ser capaz de executar, ao lado de outras duas, não menos importantes, que são: o mapeamento de ambientes e a localização. A navegação constitui-se de um

conjunto de outras sub-tarefas, das quais destacamos o planejamento de caminhos. Para planejar um caminho ou rota e ser capaz de se deslocar com segurança pelo ambiente, o robô necessitará de algumas informações como quantidade e posição de obstáculos, que foram obtidos na etapa de mapeamento e sua posição obtida na etapa de localização. A partir destas informações e por aplicação de técnicas de planejamento o deslocamento torna-se possível.

No planejamento de caminhos existem algumas técnicas já consolidadas como mapa de rotas (*roadmaps*), campos de potencial e decomposição em células. A aplicação dessas técnicas de planejamento conduzem a obtenção de grafos cujos nós representam coordenadas de pontos do ambiente e as arestas correspondem às distâncias entre estes pontos.

A escolha dos pontos que compõem a rota é um caso particular para cada abordagem, tornando necessária a aplicação de alguma técnica de busca para extrair do grafo um caminho mínino entre o nó origem e o nó destino. Neste trabalho, a técnica de inteligência artificial baseada em AG é aplicada para obtenção do menor caminho em um grafo representativo do ambiente.

O objetivo geral deste trabalho é aplicar a técnica de AG ao problema de planejamento de caminhos na abordagem de decomposição em células convexas para o caso aproximado, visando otimizar a busca por uma trajetória mínima e implementar um protótipo de robô móvel para realização de testes de execução de trajetórias, comparando os resultados com os obtidos pela aplicação do algoritmo de Dijkstra. Os objetivos específicos serão o estudo do desempenho dos algoritmos necessários para implementação de uma aplicação que permita a visualização gráfica da configuração do ambiente, contendo informações de suas dimensões, das posições dos obstáculos, dos pontos inicial e final e da trajetória obtida. Para validação prática da execução das trajetórias será implementada uma plataforma robótica simples com sistema eletrônico embarcado para acionamento diferencial das rodas e captação dos dados referentes a trajetória por meio de sinais de rádio.

Será necessário uma revisão bibliográfica dos conceitos sobre geração de trajetórias, o método de decomposição em células e aplicações de AG ao problema de busca em grafos.

Esperamos, com este trabalho, abrir portas ou pelo menos idéias para novas pesquisas dentro do estudo da Robótica e suas aplicações no PPGSCA/UFERSA.

A principal contribuição deste trabalho é a aplicação de AG ao problema de planejamento de caminhos, na abordagem de decomposição de células convexas para o caso aproximado, visando redução de tempo de processamento e complexidade da

execução desta abordagem. Também será realizada uma avaliação de desempenho pela comparação entre o algoritmo de Dijkstra aplicado e o algoritmo proposto. Como parte integrante deste estudo será concebida uma aplicação de *software* e uma plataforma robótica que permita incorporação desta técnica de geração de trajetória.

Este trabalho encontra-se estruturado da seguinte maneira:

- No **Capítulo 1** são apresentados os fundamentos teóricos e definições com ênfase em navegação de robôs móveis e algoritmos de busca em grafos.
- No **Capítulo 2** é apresentado o detalhamento do algoritmo proposto e sua aplicação ao Planejador de Caminhos.
- No **Capítulo 3** são apresentados e discutidos os resultados de simulação e experimentais.

Capítulo 1

Fundamentação Teórica e Definições

Este capítulo abordará conceitos fundamentais ao desenvolvimento do trabalho. A Seção 1.1 apresenta as principais abordagens aplicadas no planejamento de caminhos. Na Seção 1.2 são mostrados os algoritmos de busca mais utilizados em problemas de caminhos mínimos em grafos e na Seção 1.3 são apresentados conceitos básicos de AG. Finalmente, a Seção 1.4 traz uma discussão dos trabalhos mais recentes relacionados ao problema de caminhos mínimos em grafos utilizando AG.

1.1 Planejamento de Caminhos

1.1.1 Introdução

Em um ambiente povoado de obstáculos, considerando que o robô possui informações parciais ou totais da configuração deste, o planejamento de caminhos, consiste em definir uma trajetória para que este robô parta de um ponto inicial e atinja uma posição final sem colidir com os obstáculos presentes no ambiente (PEREZ; ROSELL, 2010). A trajetória ideal, quando levados em conta aspectos relativos como gasto energético, traduz-se na menor distância possível entre os pontos inicial e final. A realização desta tarefa envolve um número muito grande de variáveis, como as características cinemáticas do robô, suas dimensões e quantidade de obstáculos por exemplo. Neste contexto, são encontradas na literatura duas grandes abordagens utilizadas na definição da estratégia de planejamento de caminhos. A primeira, conhecida por planejamento local, que acontece quando o robô consegue ao longo do seu deslocamento, montar o mapa do ambiente no qual está inserido e obtiver sua localização por meio de leituras dos seus sensores. A segunda abordagem que será aplicada neste trabalho é o planejamento global, que é caracterizada quando o robô recebe a priori um mapa com as informações do ambiente e sua localização inicial (SICILIANO, 2006). Nas subseções 1.1.2 à 1.1.4 serão apresentadas cada uma

das principais abordagens utilizadas no planejamento de caminhos.

1.1.2 Mapa de Rotas

O princípio básico do mapa de rotas é a captura das características geométricas do ambiente e posteriormente a busca por um caminho livre para o robô. A técnica consiste da análise do conjunto de todos os pontos que podem se atingidos para ir da posição inicial à posição desejada, na forma de uma rede de curvas unidimensionais conhecidas por mapa de rotas (LATOMBE, 1991). O planejamento reduz-se a buscar um caminho dentro deste conjunto que interligue os pontos inicial e final. Os métodos que baseiam-se nesta abordagem são de simples implementação e dentre eles destacam-se: O Grafo de Visibilidade e Diagramas de Voronoi.

Grafos de Visibilidade

O grafo de visibilidade para o caso bidimensional de um ambiente contendo obstáculos poligonais é representado pela ligação de todos os vértices por meio de arestas, onde cada uma destas não possui intersecção com nenhuma outra que delimita o ambiente ou qualquer outra que delimita os obstáculos, não podendo também estar inteiramente contida neste. O grafo de visibilidade possui em seus nós, todos os vértices do espaço (ambiente e obstáculos) onde está inserido o robô. As arestas do grafo representam os menores caminhos entre esses vértices. A Figura 1.1 mostra um exemplo de aplicação de grafos de visibilidade para um ambiente contendo três obstáculos.

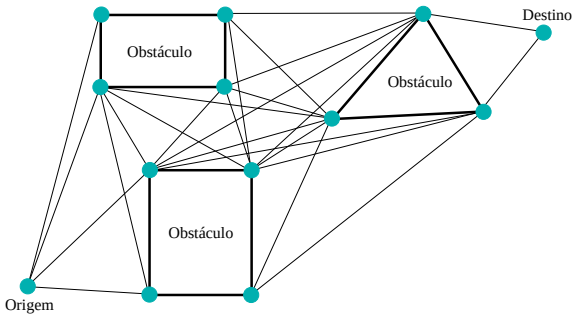


Figura 1.1: Exemplo de grafo de visibilidade
Fonte: (Autor, 2012)

O grafo de visibilidade possui características positivas por ser completo, ou seja, caso exista um caminho possível este será encontrado e também por ser ótimo,

isto é, o caminho obtido é o menor dentre os caminhos possíveis para aquele grafo de visibilidade. Entretanto, possui desvantagens em relação ao tempo de execução em consequência da alta dependência das características geométricas dos obstáculos e do ambiente, pois a cada incremento no número de vértices produz um aumento considerável no número de arestas, aumentando assim o tempo de busca. Outra característica deste método é que a rota obtida passa muito próxima aos obstáculos do ambiente, aumentando o risco de uma colisão por acúmulo de erros na medição de distâncias. Em alguns tipos de sensores utilizados pelo robô esse erro é inerente. Como exemplo podemos citar a medição de distância utilizando *encoders* incrementais.

Diagramas de Voronoi

A definição do diagrama de Voronoi diz que dado um conjunto formado por pontos denominados geradores em um plano, o diagrama de Voronoi correspondente a esses pontos que é formado pela união das subdivisões deste plano em áreas delimitadas pelos locais mais próximos a cada par de pontos geradores. Para um dado conjunto S de n pontos no plano deseja-se determinar para cada ponto p de S qual é a região $V(p)$ dos pontos do plano que estão mais próximos de p do que de qualquer outro ponto em S (KANG, 2008). As regiões determinadas por cada ponto formam uma partição do plano chamada de Diagrama de Voronoi. Em resumo pode-se dizer que a aplicação de diagramas de Voronoi a um espaço repleto de pontos resultará em um conjunto de segmentos de reta equidistantes a esses pontos. A Figura 1.2 exemplifica essa definição.

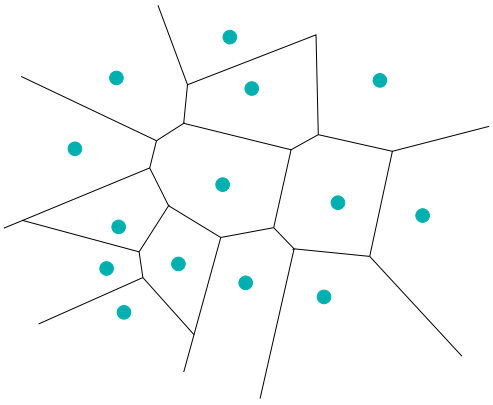


Figura 1.2: Diagrama de Voronoi
Fonte: (Autor, 2012)

As aplicações dos diagramas de Voronoi abrangem várias áreas como a astronomia, a biologia e a química, por exemplo (KANG, 2008). Na robótica, a obtenção do diagrama de Voronoi para uma área contendo obstáculos, interpretando cada vértice como sendo um ponto gerador, o diagrama traduz-se em um mapa onde cada segmento de reta representa uma rota possível. Um ponto positivo está na característica da rota ser equidistantes aos obstáculos, ou seja, o robô trafega o mais distante dos obstáculos possível. Na Figura 1.3 podemos observar a obtenção de um conjunto de rotas pela extração do diagrama de Voronoi no ambiente, onde cada vértice dos obstáculos é visto como um ponto gerador e cada aresta (pertencente a obstáculos ou ambiente) é vista como um conjunto de infinitos pontos geradores. Quando se tem uma configuração aresta-aresta ou vértice-vértice o segmento resultante é uma linha reta, e quando tem-se a configuração aresta-vértice o resultado é uma curva.

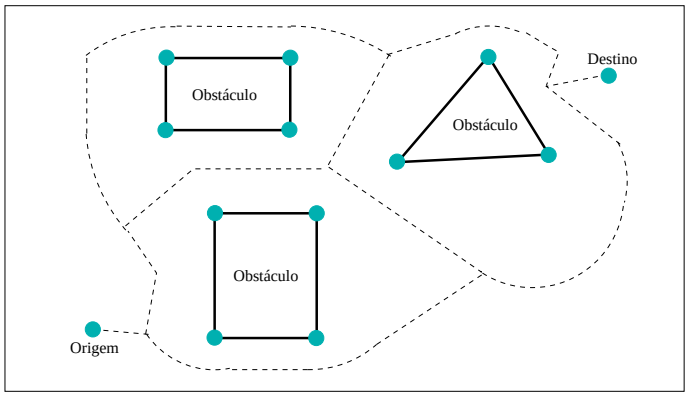


Figura 1.3: Exemplo de aplicação do diagrama de Voronoi
Fonte: (Autor, 2012)

1.1.3 Decomposição em Células

O método de decomposição em células será utilizado neste trabalho para aplicação do AG. Este método foi escolhido por ser um dos mais utilizados na geração de trajetórias em aplicações robóticas. Sua grande utilização é divido ao formato simples das células que serão formadas na decomposição do ambiente. Em linhas gerais seu funcionamento consiste em dividir, até um limite estabelecido, o ambiente em várias regiões menores chamadas de células. Toda a estrutura é transformada em um grafo onde cada nó representa uma célula e cada aresta a distância euclidiana entre uma célula e sua vizinha. O grafo é chamado de grafo de conectividade. Pela

aplicação de um algoritmo de busca ao grafo obtém-se, caso exista, um caminho mínimo entre a célula que contém o ponto inicial e a que contém o ponto final. No método de decomposição em células, duas são as abordagens existentes: A decomposição de células exata e aproximada (OTTONI; LAGES, 2000).

Decomposição Exata

Na abordagem de decomposição em células exata, tem-se a decomposição do ambiente em um conjunto de regiões não sobrepostas que ao serem unidas correspondem exatamente ao ambiente original. No processo de decomposição cada célula gerada não possui geometria predefinida e o ambiente é decomposto em regiões triangulares e trapezoidais (LATOMBE, 1991). Cada célula é obtida pelo traçado de linhas verticais partindo dos vértices dos *"C-obstáculos"* ou dos vértices que delimitam o ambiente. Um *C-obstáculo* corresponde a um aumento na área do obstáculo para que assim seja possível reduzir o robô a um ponto, o que facilita a implementação. O método torna-se mais complexo quando previr situações em que os lados dos obstáculos e/ou ambiente não formam ângulos retos entre si, o que torna mais elevado o custo de obtenção das células e verificação das relações de adjacência necessárias à construção do grafo de conectividade (SLEUMER; TSCHICHOLD-GüRMANN, 1999). A Figura 1.4 ilustra a aplicação do método utilizando linhas verticais aplicadas em cada vértice para obter a divisão em células. Cada centro geométrico de uma célula será visto como um nó no grafo de visibilidade. Para obter a trajetória, interliga-se o ponto que constitui o centro geométrico da célula até o ponto médio do segmento que divide duas células adjacentes. Esse procedimento é repetido até que o ponto de destino seja atingido, como pode ser observado na Figura 1.4.

Decomposição Aproximada

A utilização da decomposição aproximada difere do método exato, para uma mesma configuração de ambiente, pela forma geométrica das células ser mais simples e definida a priori. As formas geométricas mais utilizadas para a representação das células são quadriláteros, isso devido a facilidade de manipulação. Por esta razão temos a impossibilidade, na maioria das configurações, de uma representação exata do ambiente (OTTONI; LAGES, 2000). Dizemos então que a abordagem não é completa, diferentemente do método exato, pois a depender do limite preestabelecido para o tamanho mínimo que uma célula poderá chegar, o caminho pode não ser

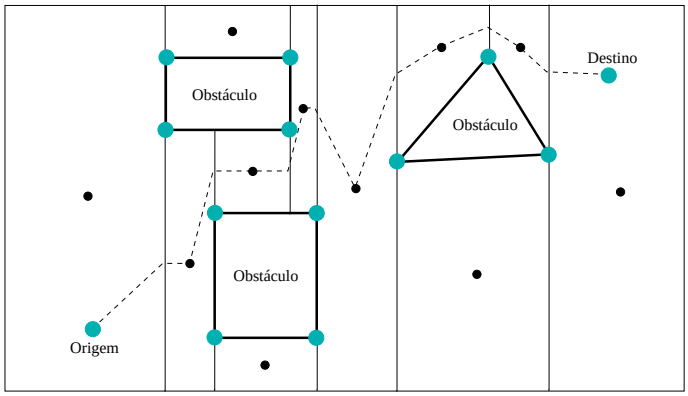


Figura 1.4: Exemplo de aplicação da decomposição em Células Exata
Fonte: (Autor, 2012)

encontrado, embora este exista. Portanto, a resolução máxima que implicará no número de células geradas no procedimento de divisão é definida previamente, pois do contrário o método produziria, a cada iteração, células cada vez menores com dimensões que tenderiam a zero. As sucessivas divisões levariam a uma quantidade de células muito elevada e consumindo todo o recurso de memória da máquina. Cada uma das células recebe um rótulo que pode ser de três tipos: Célula cheia, quando a célula está inteiramente contida em um obstáculo; Célula mesclada, quando parte da célula está contida em um obstáculo e Célula vazia, quando a célula não intersecta nenhum obstáculo, como mostra Figura 1.5.

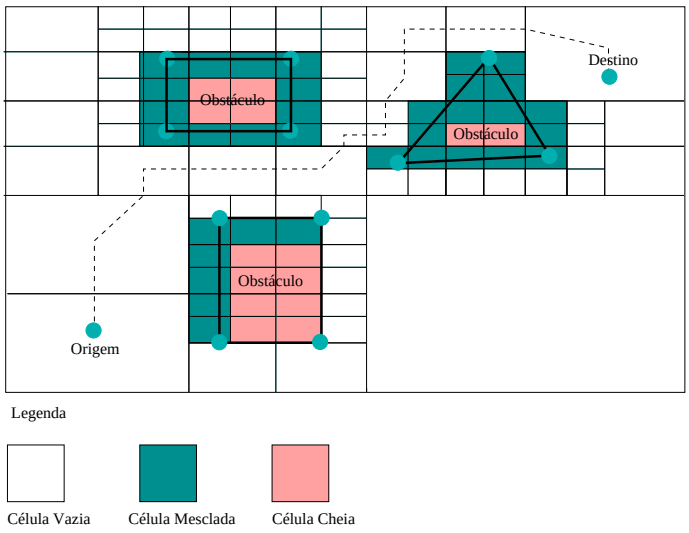


Figura 1.5: Decomposição em Células Aproximada
Fonte: (Autor, 2012)

Na execução do algoritmo apenas as células mescladas serão divididas novamente e o método pára quando o limite mínimo em relação ao tamanho da célula for atingido ou quando não existirem células mescladas a dividir.

1.1.4 Campo de Potencial

A idéia básica aplicada pelo método de campo de potencial é associar o robô a uma partícula imersa em um ambiente submetido a forças de atração e repulsão. O movimento descrito pelo robô é semelhante ao de uma bola descendo uma ladeira, onde o ponto mais elevado corresponde ao ponto de origem e o ponto mais baixo ao destino, como mostra a Figura 1.6.

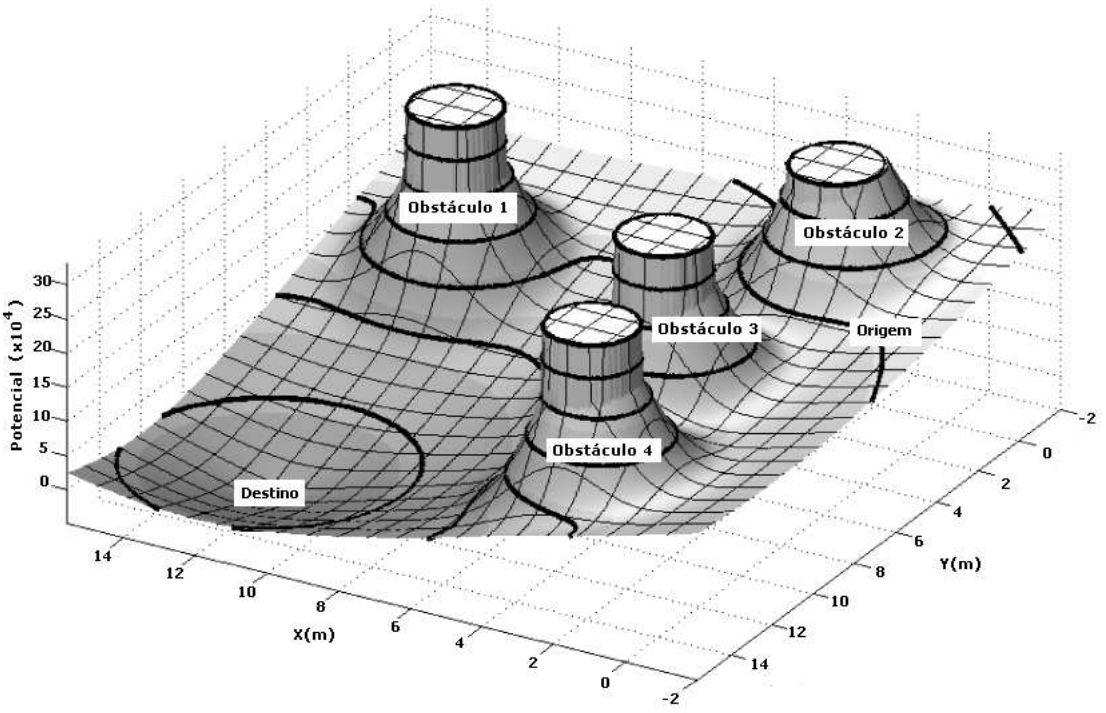


Figura 1.6: Exemplo de campo de potencial
Fonte: (CAKIR; BUTUN; KAYMAN, 2006)

O ponto destino imprimirá ao robô uma força de atração e os obstáculos forças de repulsão. Esse método é o que mais se adequa a espaços de configuração dinâmica devido a baixa complexidade exigida para o cálculo da trajetória, (SOUZA, 2008). O problema de otimização associado a aplicação de campo de potencial tem alguns gargalos de difícil contorno, como as situações de mínimos locais, ou seja, situações em que pela análise da região de vizinhança a um ponto, o método não consegue

precisar se aquele ponto é o ponto de menor potencial (mínimo global ou destino) ou se esta é uma configuração particular ainda com potencial superior ao mínimo global (situação conhecida como mínimo local). A ocorrência de mínimos locais em espaços de configuração são normalmente relacionadas com posicionamento de obstáculos em U, ou presença de corredores muito estreitos, como mostra o exemplo da Figura 1.7.

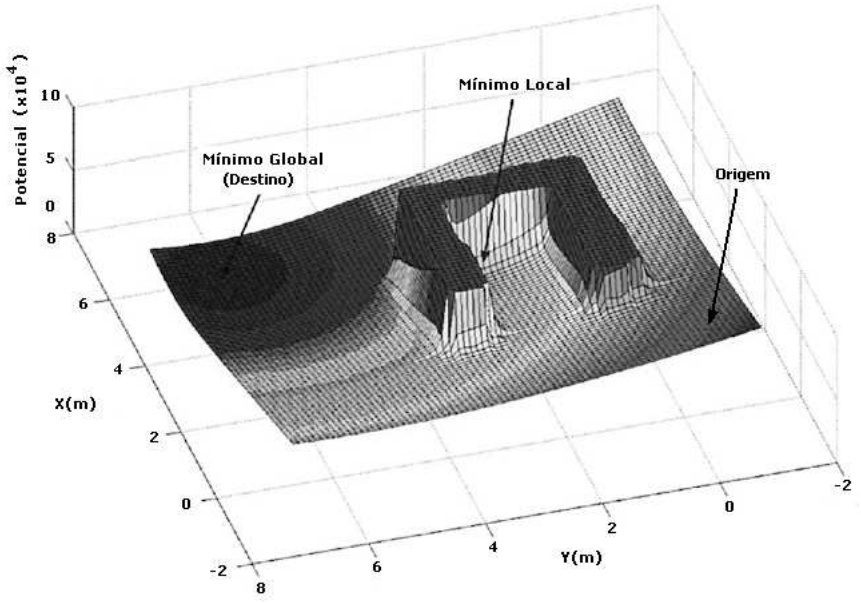


Figura 1.7: Problema de mínimo local em campo de potencial
Fonte: (CAKIR; BUTUN; KAYMAN, 2006)

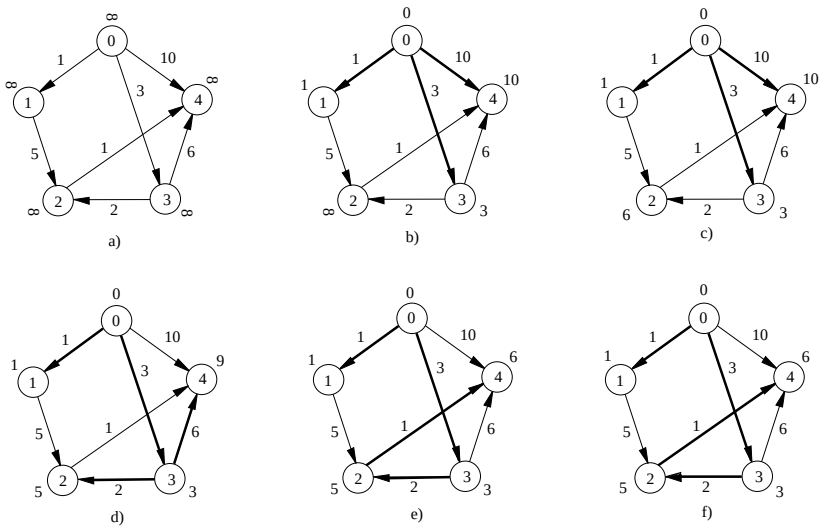
1.2 Algoritmos de busca em Grafos

No estudo dos principais métodos para planejamento de caminhos observa-se que a etapa final dos métodos recai na obtenção de um grafo com seus nós representando locais de um ambiente e as aresta as vias de conexão entre estes locais ponderadas pelo valor da distância. Para representar esta configuração faz-se normalmente uso de grafos não orientados com arestas ponderadas por valores positivos. Nesta Seção serão apresentados os principais algoritmos aplicados ao problema de busca do caminho mínimo em grafos, necessários a obtenção percurso de menor custo.

1.2.1 Algoritmo de Dijkstra

O algoritmo de Dijkstra, desenvolvido pelo holandês Edsger Dijkstra em 1956, é o mais conhecido algoritmo aplicado ao problema de busca do caminho mínimo em grafos não orientados com arestas ponderadas por pesos não negativos. O algoritmo utiliza critério guloso, ou seja, sua decisão é definida pelo ótimo naquele momento (COULOURIS; DOLLIMORE; KINDBERG, 2005).

A dinâmica do algoritmo, como mostra a Figura 1.8, consiste em encontrar os caminhos mais curtos de acordo com a distância dos nós a um dado nó origem n_o (BARROS; PAMBOUKIAN; ZAMBONI, 2007). Inicialmente são criados dois conjuntos de vértices M e Q, onde M representa todos os vértices v_i, o custo $d[v_i]$ é mínimo e Q contém os vértices abertos.



Iteração	M	d[0]	d[1]	d[2]	d[3]	d[4]
a)	$\emptyset$	∞	∞	∞	∞	∞
b)	{0}	0	1	∞	3	10
c)	{0,1}	0	1	6	3	10
d)	{0,1,3}	0	1	5	3	9
e)	{0,1,3,2}	0	1	5	3	6
f)	{0,1,3,2,4}	0	1	5	3	6

Figura 1.8: Exemplo de execução do algoritmo de Dijkstra
Fonte: (ALMEIDA; ZIVIANI, 2004)

Um vértice é dito aberto quando o caminho mínimo deste para o nó n_o ainda não tiver sido obtido. Para todos os outros nós i temos suas distâncias $d[i]$ inicializadas com um valor alto (representado por ∞). Quando todos os vértices tiverem sido

fechados, os valores obtidos para suas distâncias *d[i]* serão mínimos dos caminhos que partem do vértice n_o até os demais vértices do grafo. O caminho propriamente dito é obtido a partir da sequência de vértices chamados de precedentes. Um vértice v_{i-1} é dito precedente de um vértice v_i se para alcançá-lo faz-se necessário passar por v_{i-1} e a distância deste para n_o é a menor possível. A seguir tem-se a descrição do algoritmo de Dijkstra.

```
Entrada: Grafo, Origem
Saída: Árvore Geradora Mínima
início
    para cada vértice v em grafo faça
        distancia[v] = Infinito;
    fim
    distancia[origem] = 0;
    Q = todos os vértices de grafo;
    enquanto Q não é vazio faça
        u = vértice em Q com menor distância;
        se distancia[u] == Infinito então
            Sai do laço;
        fim
        remove u de Q;
        para cada vizinho v de u faça
            d = distancia[u] + distancia entre u e v;
            se d < distancia[v] então
                distancia[v] = d;
            fim
        fim
    fim
    devolve distancia;
fim
```

Algoritmo 1: Algoritmo de Dijkstra

No pior caso, para n igual ao número de vértices do grafo, cada iteração neste laço envolve uma pesquisa em todos os outros nós do grafo para atualização das distâncias. Assim, a complexidade do algoritmo é $O([m + n]log\ n)$, onde m é o número de arestas e n é o número de vértices.

1.2.2 Algoritmo A-Estrela

O algoritmo A-Estrela desenvolvido em 1968 por Peter Hart, Nils Nilsson, e Bertram Raphael, foi inicialmente chamado de algoritmo A. Esse é um algoritmo que utiliza critério heurístico na tomada de decisão e possui comportamento ótimo.

Foi concebido com o objetivo de reduzir o custo computacional para aplicações de busca. O critério heurístico foi integrado para guiar o procedimento de busca, estimando o custo do deslocamento da posição corrente até a posição de destino (SOUZA, 2008). Dessa forma o algoritmo reduz o tamanho da árvore de pesquisa e aumenta a velocidade do processo. O A-Estrela também possui o conceito de listas de nós abertos e fechados para realizar o controle dos nós que já foram explorados. A lista aberta possui o registros dos nós que foram alcançados, mas que ainda não foram expandidos, como mostra a Figura 1.9, a lista fechada contém o registro dos nós que já foram visitados e expandidos. O algoritmo A-Estrela é completo, ou seja, ele sempre encontrará o menor caminho, se esse existir. Embora a função que define a heurística seja inexata. Em contrapartida a sua complexidade em tempo e uso de memória são $O(n^2)$.

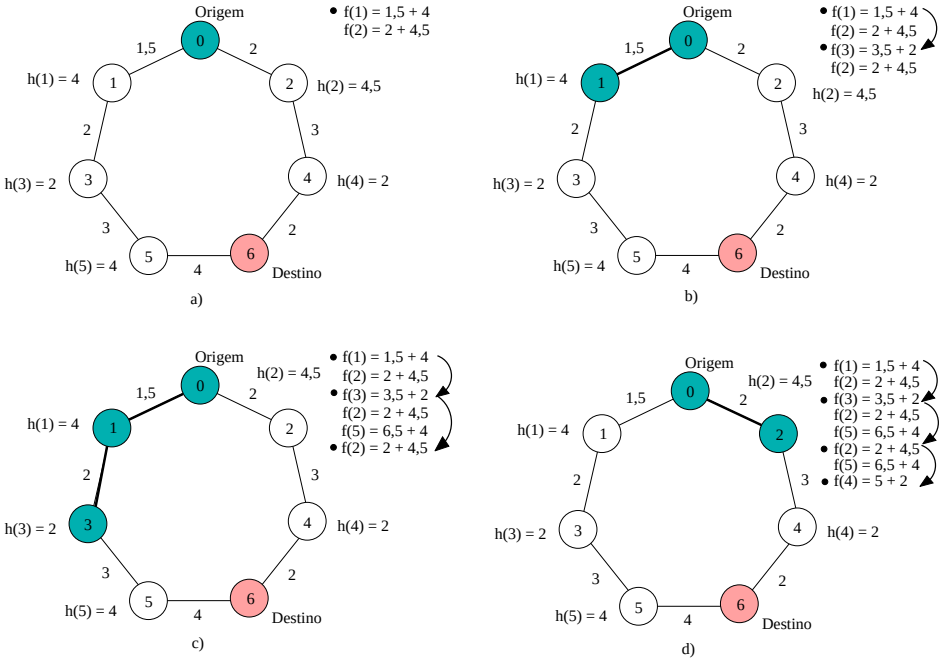


Figura 1.9: Exemplo de execução do algoritmo A-Estrela
Fonte: (Autor, 2012)

```
Entrada: Grafo, Origem, Destino
Saída: Caminho Mínimo
início
    Adicionar o vértice Origem a lista de ABERTOS enquanto ABERTOS
    não é vazio faça
        vértice corrente = vértice em ABERTOS com a menor distância;
        se vértice corrente == destino então
            Sai do laço;
        fim
        remove u de Q;
        para cada vizinho vértice corrente faça
            se vizinho ∈ a lista FECHADOS então
                Ignore;
            fim
            senão
                se vizinho ∉ a lista ABERTOS então
                    Guardar custo g e estimativa h de vizinho;
                fim
                se vizinho ∈ a lista ABERTOS então
                    Confere se o caminho é menor usando o custo g como
                    medida; Caso seja menor recalcula-se o custo g e a
                    estimativa h;
                fim
            fim
        fim
    fim
    Se encontrou Destino, utiliza a lista de seus antecedentes
    para refazer o menor caminho até Origem
fim
```

Algoritmo 2: Algoritmo A-Estrela

1.3 Algoritmos Genéticos

Algoritmos Genéticos (*GA's-Genetic Algorithms*) constitui-se na técnica computacional inteligente de busca e otimização, inspirada em conceitos biológicos e na teoria *Darwiniana* da evolução (GOLDBERG, 1989). Essa técnica foi originalmente desenvolvida pelo americano John Henry Holland's, onde mecanismos naturais de seleção e reprodução genética (cruzamento e mutação) foram formalizados incorporados ao algoritmo. As soluções para um determinado problema são codificadas através de cromossomos. Os cromossomos são estruturas de dados que representam possíveis soluções do espaço de busca do problema. Os cromossomos por

sua vez são submetidos a processos que envolvem a avaliação de sua aptidão, seleção, cruzamento e mutação. Depois de atingido o critério de parada a população deverá conter indivíduos mais aptos representando as melhores soluções para o problema. A Figura 1.10 representa em diagrama de blocos as etapas do algoritmo.

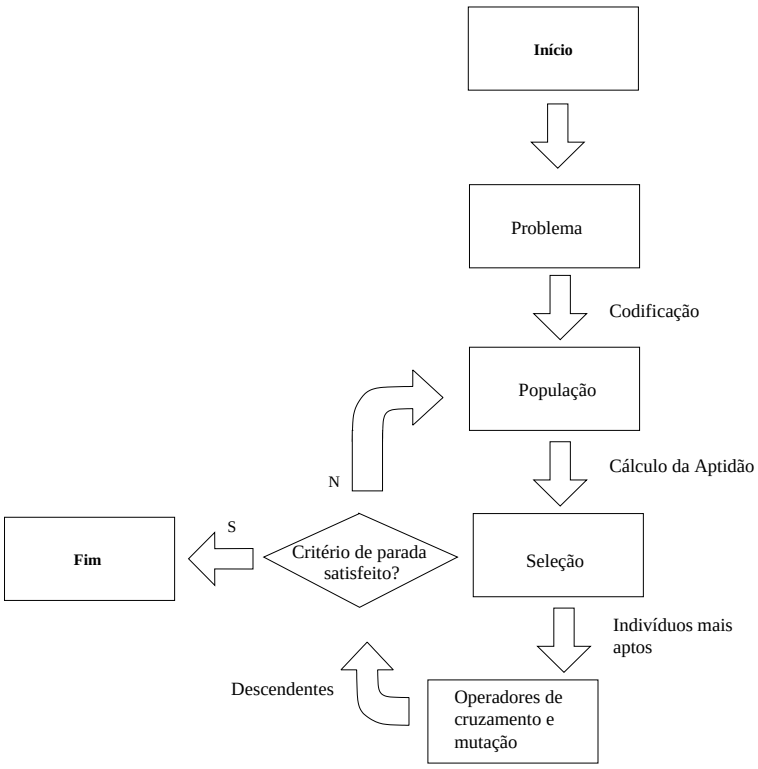


Figura 1.10: Fluxograma de execução do Algoritmo Genético
Fonte: (Autor, 2012)

1.3.1 Representação dos cromossomos (Codificação)

A forma como são representadas as soluções do espaço de busca de um problema define como será montado o cromossomo. Essa representação depende do tipo de problema a que se aplicará o algoritmo. Existem várias maneiras de representar cromossomos, as mais utilizadas são: a binária, número inteiros e símbolos, ver Figura 1.11. A representação binária é a mais utilizada pela facilidade de manipulação (PACHECO, 1999).

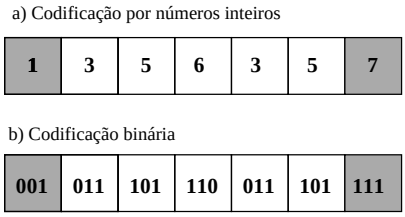


Figura 1.11: Exemplos de representação de cromossomos
Fonte: (Autor, 2012)

1.3.2 Inicialização da população

A inicialização da população consiste na criação dos primeiros indivíduos para o início da execução do algoritmo. Normalmente a geração desses indivíduos acontece de forma aleatória, mas também poderá ser guiada por uma heurística para melhorar o número de indivíduos viáveis e/ou de boa aptidão. Cada heurística aplicada depende intimamente do problema em questão e poderá ser bastante útil para gerar um número alto de indivíduos que representem boas soluções.

1.3.3 Função de avaliação

A etapa de avaliação torna-se um ponto chave do algoritmo, pois através dela é que o cromossomo é pontuado. A função necessita representar muito bem o problema fazendo com que boas soluções possam levar o algoritmo a convergência para o ótimo resultado.

1.3.4 Seleção de Indivíduos

Nesta fase os indivíduos com melhor aptidão, ou seja, os indivíduos melhor avaliados pela função de avaliação possuem maior probabilidade de serem escolhidos para a reprodução. Na literatura, são encontrados cinco principais mecanismos de seleção: proporcional, por torneios, com truncamento, por normalização linear e por normalização exponencial (PACHECO, 1999). Um mecanismo de seleção possui a característica de intensificar a variação na aptidão média da população, permitindo que a população evolua sem perder a diversidade do material genético o que poderia levar o algoritmo a um mínimo local.

1.3.5 Operadores Genéticos

Os indivíduos selecionados serão recombinados pelo operador genético de cruzamento mediante um valor de probabilidade. A maneira como o material genético dos genitores será combinada para formar um novo indivíduo é particular de cada problema. De maneira geral essa combinação acontece pelo corte dos cromossomos pais em um ou mais pontos. A formação do descendente é, então, realizada pela mescla das partes dos cromossomos dos genitores provenientes do corte. O operador de mutação é aplicado também mediante uma probabilidade. No geral o valor da probabilidade é baixo para evitar uma grande diversificação da população.

1.4 Trabalhos Relacionados

Esta Seção discute os principais trabalhos diretamente relacionados com a aplicação de técnicas inteligentes ao problema de busca em grafos. Serão apontados alguns aspectos relevantes, e úteis ao desenvolvimento do algoritmo proposto. Apesar de não terem sido encontrados na literatura aplicações de AG em problemas de grafos gerados por técnicas de planejamento de caminhos, foram localizados alguns trabalhos que utilizam a técnica para resolver problemas de menor caminho em redes de computadores. A maioria dos trabalhos modela, por AG, situações de tráfego de pacotes em sistemas de interligação de roteadores.

O trabalho de (NAGIB; ALI, 2010) apresenta o desenvolvimento de um AG para resolver o problema de protocolo de roteamento em redes buscando o caminho mais curto entre a fonte e o nó de destino. Seu algoritmo utiliza como função custo a distância física entre os roteadores e os cromossomos utilizados em sua implementação possuem tamanho fixo. O cruzamento é realizado apenas em um ponto de corte no cromossomo e o operador de mutação consiste na troca aleatória de dois genes. Em seu trabalho (NAGIB; ALI, 2010) apresenta experimentos realizados em grafos com um número muito pequeno de nós, configurações com máximo 10, o que não se aplica aos grafos obtidos pelo método de decomposição em células convexas para caso aproximado, que facilmente pode atingir valores maiores.

O trabalho de (AHN; RAMAKRISHNA, 2002) apresenta uma abordagem com maiores refinamentos. Os cromossomos possuem um tamanho variável, e o cruzamento, também de um ponto de corte, avalia se os pais possuem genes em comum. O corte é realizado exatamente nesses genes, evitando a gerar filhos

que representem caminhos inválidos. Outro ponto importante em seu trabalho foi a introdução de uma função de remoção de laços (*loop's*) que extrai partes do cromossomo entre dois genes repetidos. (AHN; RAMAKRISHNA, 2002) discute ainda a influência do tamanho da população no desempenho do algoritmo. Seus resultados mostram o desempenho para redes com 15 a 50 nós e a comparação do desempenho com o algoritmo de Dijkstra e os algoritmos desenvolvidos por (MUNEMOTO; TAKAI; SATO, 1998) e (INAGAKI; HASEYAMA; KITAJIMA, 1999).

Em seu trabalho, (MUNEMOTO; TAKAI; SATO, 1998) propõe uma versão adaptada na operação de cruzamento utilizada por (AHN; RAMAKRISHNA, 2002) que consiste na identificação de potenciais pontos de cruzamento, pois nos AG de roteamento, o processo de simplesmente combinar partes dos cromossomos pais leva facilmente a inconsistência (filhos inviáveis), pois o trechos a serem combinados dependem de sua localização.

(INAGAKI; HASEYAMA; KITAJIMA, 1999) apresentou um algoritmo muito semelhante ao trabalho de (MUNEMOTO; TAKAI; SATO, 1998), porém, considerando como solução do algoritmo, além da rota mínima, rotas sub-mínimas alternativas. O que não é possível obter pela aplicação do algoritmo Dijkstra, por exemplo.

Para obter melhores resultados, alguns trabalhos unem características de mais de um algoritmo. No trabalho de (ZENG; ZHANG; WEI, 2011) propõe-se um algoritmo híbrido genético-A*. A utilização do método A* visa otimizar a busca evitando explorar nós não promissores como ocorre com o algoritmo Dijkstra e tarefa básica do AG, de auxiliar o procedimento de otimização pelo uso dos operadores de cruzamento e mutação.

Capítulo 2

Algoritmo Proposto

Nesse capítulo é descrita a técnica de Algoritmos Genéticos aplicada a resolução do problema de planejamento de caminhos com a abordagem de Decomposição em Células Convexas Aproximadas com as particularidades das implementações.

2.1 Caracterização do Problema

O problema de planejamento de caminhos pode ser resolvido por diversas técnicas apresentadas no capítulo 1, todas elas recaem na busca pelo menor percurso evitando os obstáculos no intuito de minimizar os gastos de energia e tempo, podendo ser visualizadas como um problema de otimização (SOUZA, 2008). No caso específico da Decomposição em Células Convexas, objeto deste estudo, a parte final da execução do método recai na busca em um grafo onde os nós representam as células e as arestas as distâncias euclidianas entres estas. A aplicação do AG visa diminuir, na média, o custo computacional pelo uso de uma metodologia inteligente em comparação com o uso de critérios gulosos como é o caso do algoritmo de Dijkstra utilizado com maior frequência.

2.2 Algoritmo de Decomposição em Células

O algoritmo de decomposição em células consiste na decomposição do ambiente em regiões não superpostas denominadas células. Essas células possuem geometria simples cuja união é uma aproximação conservadora do ambiente. Para execução do procedimento de decomposição é necessário a construção de um grafo de conectividade G que representa as relações de adjacência entre as células. Em seguida é feita a busca de um canal que representa uma sequência de células adjacentes ligando a célula contendo o ponto inicial q_{ini} à célula contendo o ponto final q_{fim}.

No algoritmo, o robô é visto como um único ponto localizado em seu centro geométrico. Para compensar essa redução faz-se necessário aumentar as dimensões do obstáculo que passa a ser chamado de *C-obstáculo*. O primeiro passo na implementação do algoritmo é o de transformar cada obstáculo em seu *C-obstáculo* correspondente. O Cálculo formal para obtenção de um *C-obstáculo* leva em consideração a pose (posição e orientação) do robô apenas para um determinado instante, sendo necessário o cálculo de um novo *C-obstáculo* cada vez que a pose do robô fosse alterada em sua trajetória. Como consequência, o algoritmo de decomposição em células deveria ser novamente aplicado, o que elevaria o custo computacional.

A Figura 2.1 ilustra a influência da orientação do robô na obtenção do *C-obstáculo*. Na Figura 2.1 a), o robô R possui orientação θ_0 com origem no ponto O_R. Na Figura 2.1 b) há uma pequena variação de θ_0 para θ_1 resultando na mudança na forma do *C-obstáculo* obtido.

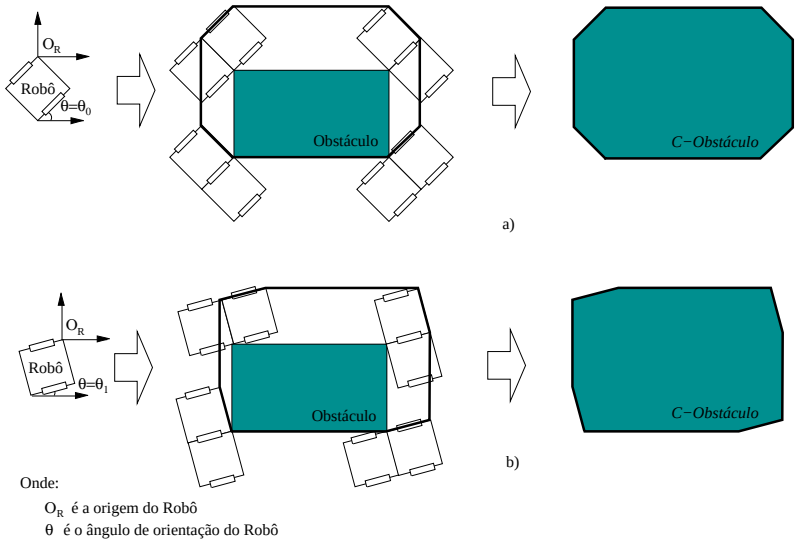


Figura 2.1: Influência da orientação do robô na obtenção do *C-obstáculo*
Fonte: (adaptado de (ALSINA, 2001))

Para contornar este problema considerando que o robô utilizado tem geometria quadrada, procedemos a geração dos *C-obstáculos* uma única vez, aumentando em cada vértice original dois novos vértices deslocados o equivalente à máxima distância entre o centro do robô e sua borda mais externa. Essa simplificação mostrada na Figura 2.2 possui a desvantagem de que, em configurações com obstáculos muito próximos, o caminho, mesmo que possível, pode não ser encontrado.

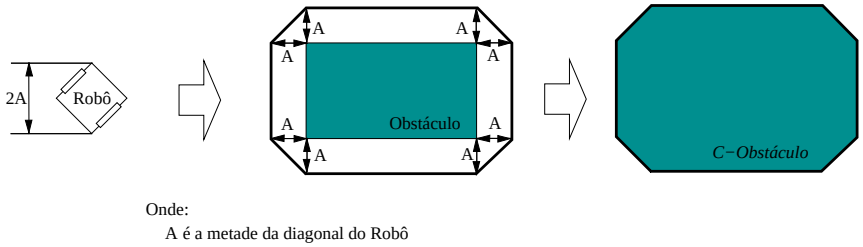


Figura 2.2: Simplificação para obtenção do *C-obstáculo*
Fonte: (Autor, 2012)

O passo seguinte no método de decomposição em células é o procedimento de divisão e rotulagem. Nesta etapa o ambiente completo é visto como uma única célula que deverá ser classificada em um dos três tipos mostrados na Figura 2.4. Nesta fase torna-se necessário a implementação de rotinas para teste de colisão de polígonos (Figura 2.3), ou seja, para verificar se a célula em teste está totalmente contida em um obstáculo, considera-se a célula cheia, parcialmente contida, considera-se a célula mesclada ou sem colisão célula livre. O teste consiste em avaliar se cada ponto correspondente a um vértice da célula faz intersecção com os *C-obstáculos*. Portanto considera-se uma célula C_i cheia se $C_i \subseteq CB$, onde CB é o conjunto dos *C-obstáculos*. C_i é considerada mesclada para $C_i \cap CB \neq \emptyset$ e $C_i \cap C_L \neq \emptyset$, onde C_L é o espaço livre. E vazia quando $C_i \cap CB = \emptyset$.

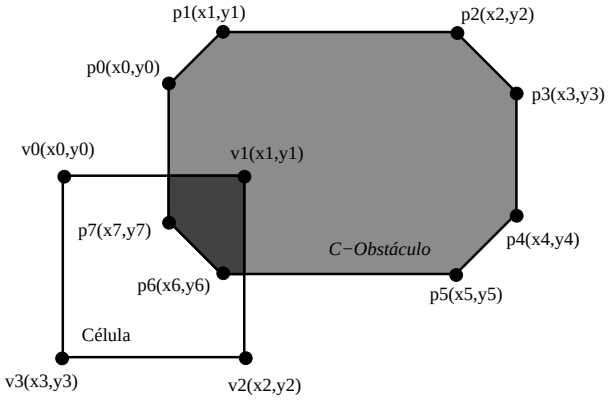


Figura 2.3: Teste de colisão para classificação das células
Fonte: (Autor, 2012)

As células cheias e vazias não serão mais analisadas. As mescladas serão divididas igualmente em quatro células filhas e em cada uma a classificação é repetida. O procedimento é recursivo até que o limite de tamanho mínimo admissível para

a célula seja atingido. O resultado desta execução é uma árvore onde a raiz corresponde ao ambiente inteiro, e em cada nível, para cada pai, serão vinculados quatro filhos resultados de sua divisão, como mostra a Figura 2.4.

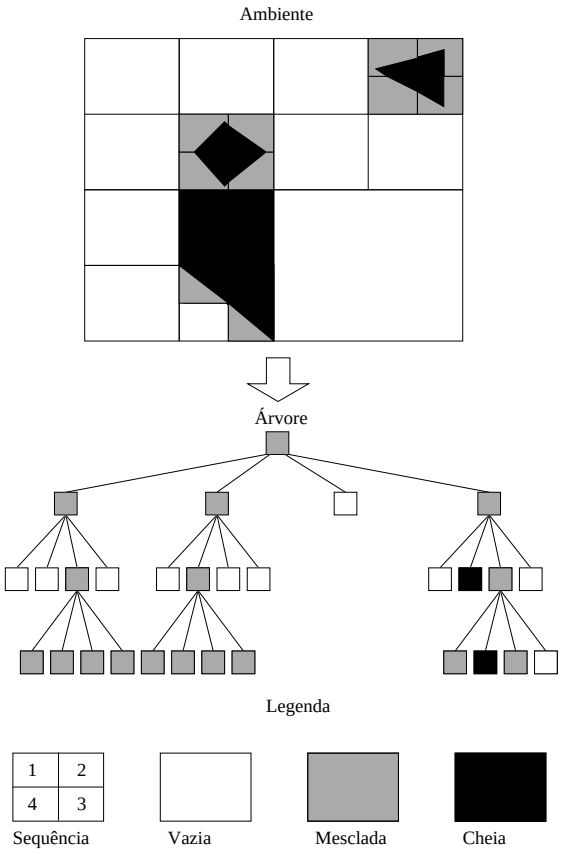


Figura 2.4: Obtenção da árvore de células
Fonte: (adaptado de (ALSINA, 2001))

A partir da árvore obtida na etapa anterior, é necessário construir o grafo de conectividade G que interliga apenas células vazias. Para obtenção do grafo de conectividade é necessário realizar um teste nas arestas das células livres para encontrar a relação de vizinhança entre as células. Este teste consiste em verificar se dois segmentos de reta de células distintas possuem pontos em comum, caso isso ocorra, a relação é armazenada em um vetor para a célula em teste. Ao mesmo tempo em que se confirma uma vizinhança, calcula-se a distância euclidiana entre os centros das células vizinhas, que é utilizado no grafo de conectividade como valor de ponderação da aresta que liga as células vizinhas. A última etapa antes da aplicação de um método de busca no grafo é a identificação dos nós de origem e destino que são obtidos pela comparação das coordenadas lidas do arquivo de dados

e os vértices que delimitam as células livres. O resultado do processamento pode ser visto na Figura 2.5.

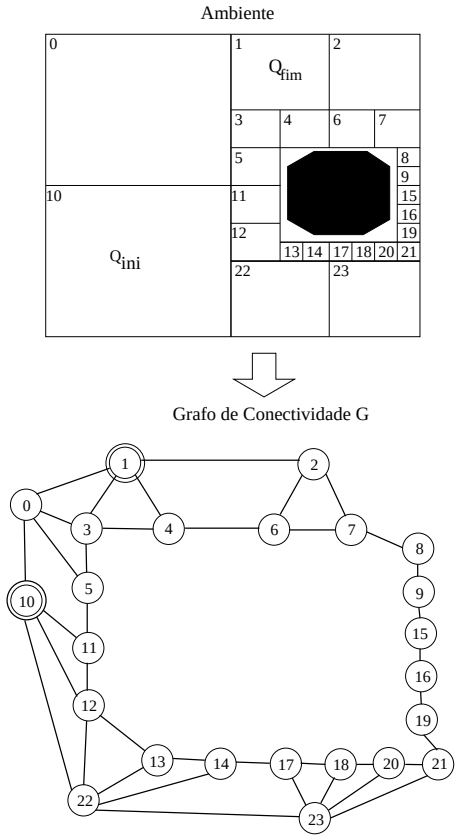


Figura 2.5: Exemplo de configuração de ambiente e seu respectivo grafo de conectividade
Fonte: (Autor, 2012)

2.3 Algoritmo Genético Proposto

O algoritmo proposto, tem por objetivo extrair do grafo de conectividade, obtido a partir da aplicação do algoritmo de decomposição em células convexas para o caso aproximado, um caminho mínimo entre os nós de origem e destino. O critério de otimização baseia-se na técnica heurística evolutiva de algoritmos genéticos. O custo ou peso da aresta que interliga dois nós corresponde a distância euclidiana entre duas células vizinhas. Neste contexto, cada cromossomo representa um caminho contendo n genes e o seu custo associado será a soma dos custos do gene i ao gene $i+1$ para i variando de 1 até n. Através da aplicação dos operadores de seleção, cruzamento e mutação os melhores indivíduos (cromossomos de menor custo) serão recombinados

para geração de uma nova população (GOLDBERG, 1989).

Para executar o algoritmo, procede-se com a inicialização dos cromossomos com os pontos inicial e final, geração da população inicial, cálculo da aptidão de cada indivíduo, seleção dos indivíduos mais aptos, cruzamento entre os indivíduos aptos para geração de uma nova população e a mutação de indivíduos. Nas subseções 2.3.1 à 2.3.6, que seguem serão detalhadas cada etapa do AG proposto.

2.3.1 Inicialização dos Cromossomos

Cada cromossomo é formado por um conjunto M de genes, onde M é um vetor com posições que variam de 2 à N, sendo N o número total de nós do grafo de conectividade, ver Figura 2.6 a). Cada gene é representado por um número inteiro positivo correspondente a um nó do grafo de conectividade G. Deste modo, cada cromossomo constitui um caminho entre o ponto inicial e final. A inicialização dos cromossomos é feita fixando-se em seu primeiro e último genes os nós que contém os pontos inicial e final, respectivamente, como mostra a Figura 2.6 b). O tamanho dos cromossomos é variável e, no pior caso, o caminho mínimo válido conterá todos os nós do grafo.

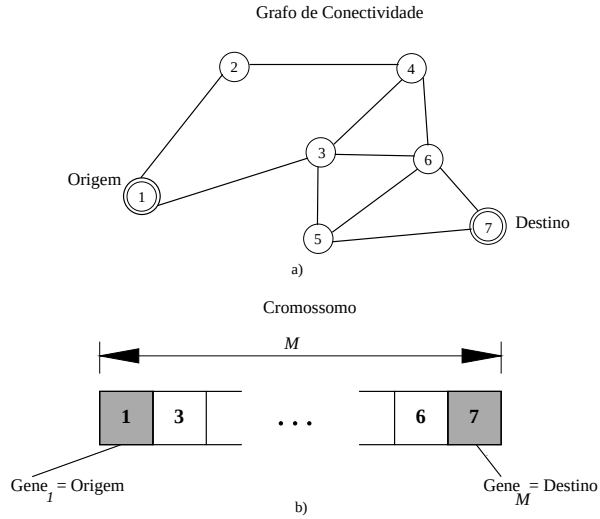


Figura 2.6: (a) Grafo de conectividade e (b) exemplo de cromossomo
Fonte: (Autor, 2012)

2.3.2 População Inicial

A geração dos indivíduos que formarão a população inicial pode ser realizada de maneira aleatória ou heurística. Nesta fase foram testadas duas técnicas. Inicialmente, optou-se por fixar os genes inicial e final, sorteando aleatoriamente os demais e mantendo o tamanho do cromossomo com um valor N fixo correspondendo ao número total de nós do grafo.

Após os primeiros testes, observou-se um número muito baixo de indivíduos viáveis e que a repetição de genes inevitavelmente conduzia à formação de caminhos com laços. Para resolver este problema, passamos a utilizar cromossomos de tamanho variável, que eram finalizados quando o gene de destino era sorteado, como mostra a Figura 2.7 b). Para aplicação deste procedimento, é necessário também a utilização de uma função de ajuste para remoção de eventuais laços, mostrada com detalhe na Seção 2.3.3.

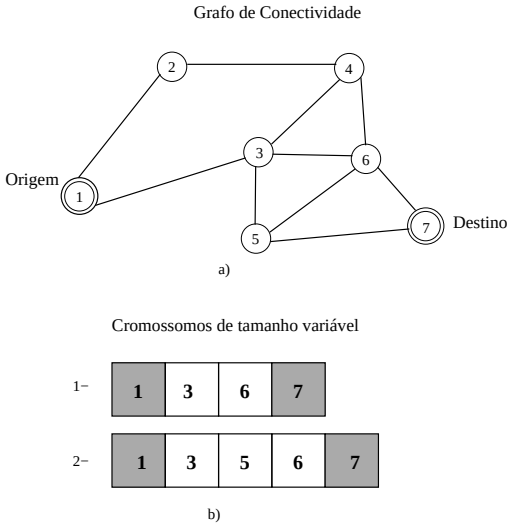


Figura 2.7: (a) Grafo de conectividade e (b) exemplo de cromossomos de tamanho variável
Fonte: (Autor, 2012)

Novos testes mostraram que a geração totalmente aleatória de cromossomos para o problema de representação de caminhos não apresenta bons resultados, pois a medida que o número de nós do grafo de conectividade G aumentam, torna-se cada vez mais difícil a obtenção de uma sequência válida entre os genes de início e fim, aumentando muito o número de indivíduos inviáveis em relação aos indivíduos viáveis.

Os indivíduos viáveis são aqueles onde existe pelo menos uma aresta interligando

dois nós representados por dois genes vizinhos, isto é, o cromossomo representa um caminho possível entre a origem e o destino pela sequência em que seus genes ocorrem. Caso o caminho não seja possível atribui-se ao indivíduo que representa esse caminho a condição de inviável. A Figura 2.8 b) mostra um cromossomo inviável, devido a inexistência de uma aresta que interligue os nós 2 e 3.

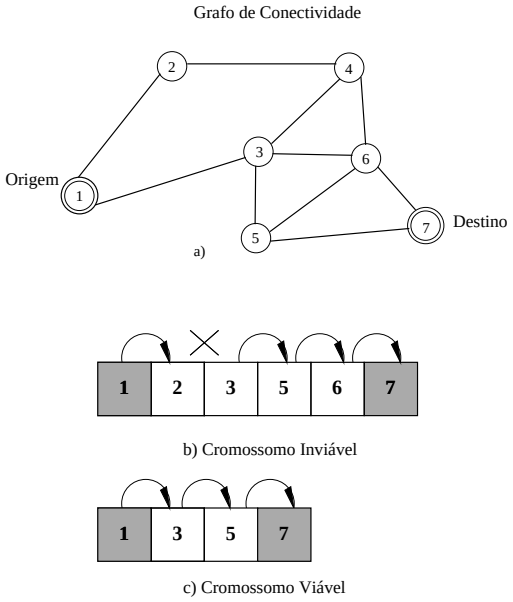


Figura 2.8: (a) Grafo de conectividade e (b) exemplo de cromossomos viáveis e inviáveis
Fonte: (Autor, 2012)

A esparsidade é outra característica dos grafos obtidos pelo método de divisão em células. Um grafo é dito esparso quando o número de arestas é próximo ao número de vértices (DIESTEL, 2010). No grafo de conectividade a esparcidade ocorre pela presença de um obstáculo. Através do grafo de conectividade da Figura 2.5 pode-se observar que o obstáculo reduz o número de arestas das células que encontram-se na sua periferia. Essa redução pode ocasionar a formação de corredores que na grande maioria das situações facilita o procedimento de busca mas pode levar algoritmos gulosos a busca em regiões pouco promissoras pela formação de corredores sem saída. Esse detalhe é mostrado na Figura 2.9.

Dessa forma, ao invés de sortear aleatoriamente um nó do grafo para compor o próximo gene do cromossomo, sorteia-se um dos vizinhos do gene corrente para obter o próximo gene. O cromossomo é finalizado quando o gene destino é sorteado, ou o tamanho do cromossomo atinge o número total de vértices do grafo. Caso o último gene do cromossomo não corresponda ao nó de destino o cromossomo é considerado

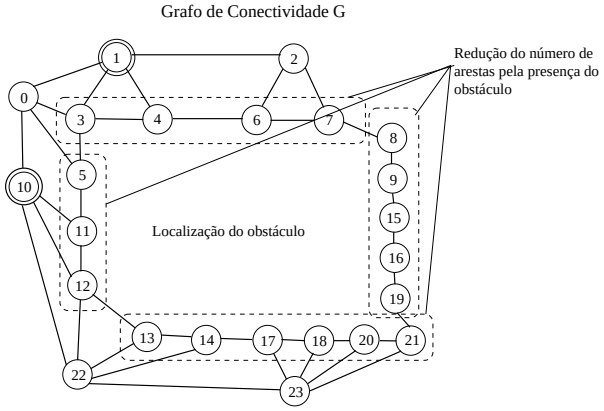


Figura 2.9: Característica esparsa dos grafos de conectividade
Fonte: (Autor, 2012)

inviável, como pode ser visto na Figura 2.10. Aqui, essa técnica heurística é utilizada para otimizar o número de indivíduos viáveis.

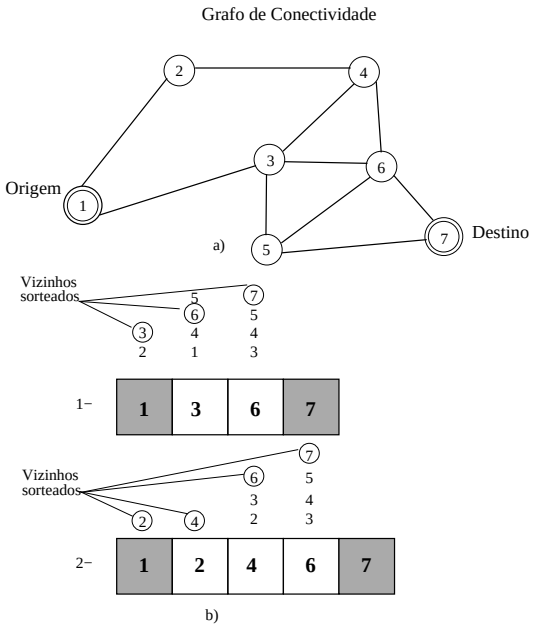


Figura 2.10: (a) Grafo de conectividade e (b) exemplo de geração heurística dos cromossomos

Fonte: (Autor, 2012)

2.3.3 Função de Ajuste

A função de ajuste é uma rotina que é aplicada após cada geração de indivíduos ou após as etapas de cruzamento e mutação. Essa função tem por finalidade procurar genes repetidos em um cromossomo e removê-los. Esse procedimento remove laços e corrige cromossomos que se tornaram inviáveis devido a presença de genes que não mantêm relação de vizinhança entre dois genes repetidos, como ilustrado na Figura 2.11.

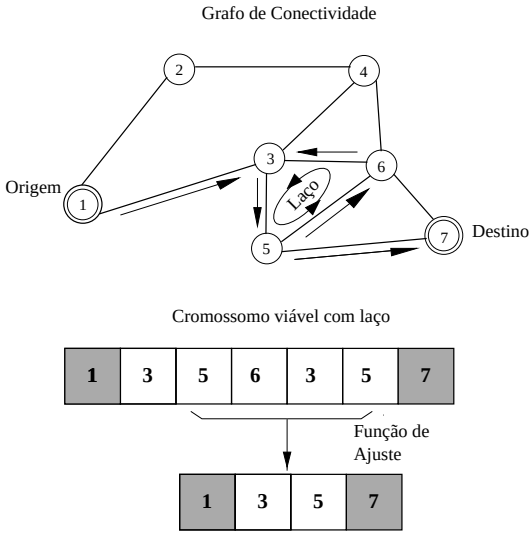


Figura 2.11: Exemplo de remoção de laço pela aplicação da função de ajuste
Fonte: (Autor, 2012)

2.3.4 Cálculo da aptidão dos indivíduos

Nessa etapa, cada indivíduo gerado é submetido a um método de verificação de sua aptidão. Aqui, a escolha de uma função que valorize a aptidão dos indivíduos definirá o sucesso na convergência do algoritmo. As características desta função são bastante específicas ao problema no qual o algoritmo foi aplicado (NAGIB; ALI, 2010), (AHN; RAMAKRISHNA, 2002). Para o problema de busca do caminho mínimo em grafos, torna-se necessário a verificação da viabilidade dos indivíduos, avaliando se o caminho indicado pelo cromossomo é válido. O objetivo dessa verificação é saber se é possível atingir um gene G_2 a partir de um gene G_1, isto é, se o nó representado por G_2 é adjacente ao nó representado por G_1. Caso esta condição

seja satisfeita, verifica-se se G_3 é alcançável por G_2 e assim sucessivamente, até que o último gene G_M do cromossomo seja alcançado. O custo total do cromossomo é o somatório dos custos entre um gene G_i e seu vizinho G_{i+1}, onde i varia de 1 ao penúltimo gene $n-1$, representado pela distância euclidiana entre os nós que estes genes representam, como mostra a Figura 2.12.

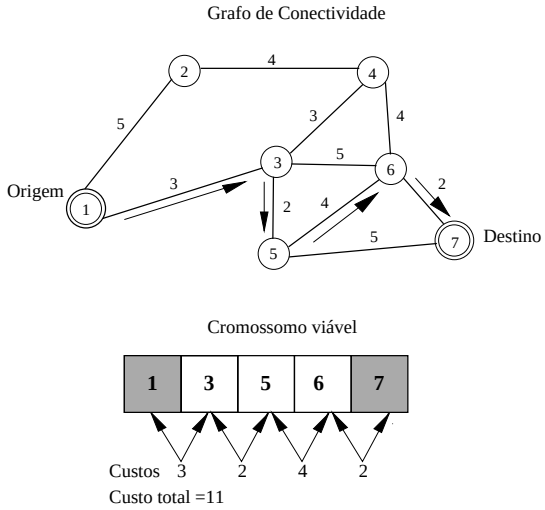


Figura 2.12: Exemplo de cálculo do custo de um cromossomo viável
Fonte: (Autor, 2012)

A função de aptidão retornará um valor cada vez maior quanto menor for o custo do cromossomo. A aptidão é definida pela equação (2.1).

$$F_a = \begin{cases} \dfrac{1}{\sum_{i=1}^{M-1} C_i(G_i, G_{i+1})} & ; \text{ Indivíduo viável} \\ 0 & ; \text{ Indivíduo não viável} \end{cases} \tag{2.1}$$

onde $C_i(G_i, G_{i+1})$ representa o custo entre o gene G_i e G_{i+1} e M o número total de genes do cromossomo.

2.3.5 Seleção dos Indivíduos Aptos

Existem várias maneiras de selecionar indivíduos em uma população (SIVARAJ; EVERETT, 2011) e a principal finalidade desta etapa é permitir que bons genes

sejam transmitidos as próximas gerações e impedir a transferência de genes que não permitirão evolução da população para o resultado ótimo. A separação dos indivíduos baseia-se no valor da função de aptidão. As etapas de seleção e cruzamento são críticas no que se refere ao desempenho do algoritmo, pois devem ser suficientes para produzir a solução ótima fugindo dos mínimos locais. O correto ajuste da taxa de cruzamento garante a diversidade da população, mantendo baixo o custo computacional para não inviabilizar o uso do algoritmo em detrimento de outras soluções.

Com base no valor da aptidão de cada indivíduo, obtida através da equação (2.1), são selecionados os melhores cromossomos da população. Caso nenhum pai viável tenha sido obtido da população inicial (que é bastante comum, dependendo do número de nós do grafo de conectividade, devido sua esparsidade para o problema de decomposição em células), uma nova população é aleatoriamente gerada com M indivíduos. Caso apenas um pai viável tenha sido obtido, é gerada uma nova população com $M-1$ indivíduos. Se mais de um pai viável for obtido será realizado o cruzamento dois a dois sorteados segundo o método de seleção escolhido. Dessa forma, a equação (2.2) é usada para que a quantidade de indivíduos gerados mais os pais atinja o limite máximo M da população, e as novas populações mantenham-se sempre com o mesmo número de indivíduos.

$$Nc = \begin{cases} \lfloor Np/2 \rfloor & ; \quad se \quad 2 \times Np \geq P \\ 2 \times Np & ; \quad caso \quad contrário \end{cases} \tag{2.2}$$

onde Nc representa o número de cruzamentos, Np o número total de pais participantes da seleção e P a quantidade máxima de indivíduos da população.

Nos métodos mais tradicionais de seleção incluem-se: proporcionais, classificação e torneio. Nos métodos proporcionais temos que a escolha dos pais será proporcional aos valores das aptidões em relação a soma total destas ou em comparação com a média retirada de todas as aptidões dos indivíduos submetidos a seleção, a roleta é o mais conhecido desses métodos. Cada fatia da roleta é proporcional ao valor da aptidão de cada cromossomo e quanto maior for esse valor, mais larga será a fatia. A Figura 2.13 mostra um exemplo de construção da roleta para um dado conjunto de cromossomos.

Pela seleção por classificação os indivíduos são ordenados por valor decrescente de aptidões e sua posição de classificação permite a associação de um valor percentual utilizado para escolha dos pais, essa técnica evita o problema que ocorre no método da roleta para populações onde a diferença de aptidão entre o indivíduo mais apto e

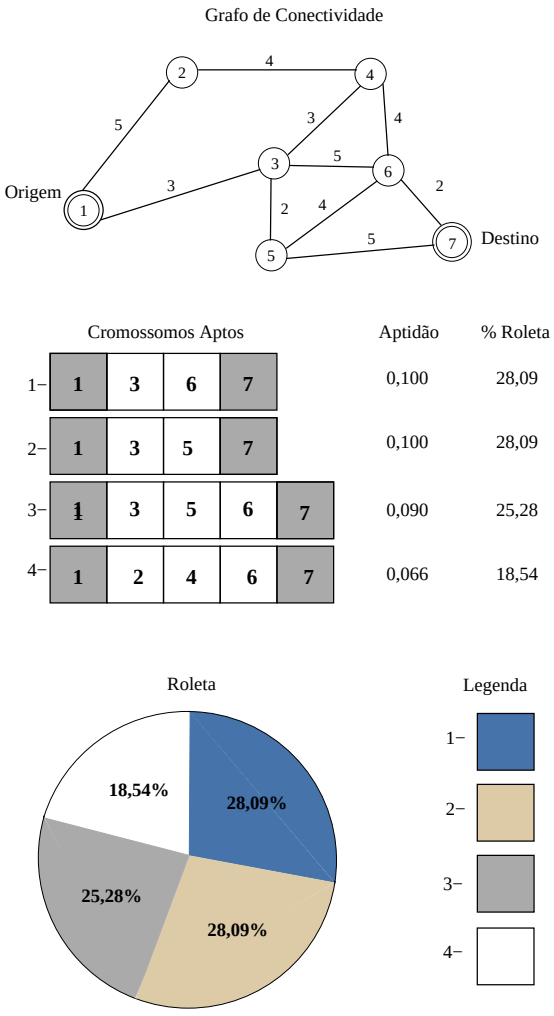


Figura 2.13: Exemplo da construção da roleta a partir de uma lista de cromossomos aptos
Fonte: (Autor, 2012)

os demais seja muito grande. E na seleção por torneio, uma competição de indivíduos selecionados ao acaso leva a escolha do indivíduo com maior aptidão.

Da forma como os indivíduos foram modelados e devido o problema possuir a característica de não haver diferenças significativas entre o cromossomo mais apto e os demais, optou-se por utilizar o método da roleta na etapa de seleção. A implementação consiste inicialmente na soma de todos os valores de aptidões dos indivíduos aptos, o que corresponde a 100% da área da roleta.

Posteriormente sendo feito o cálculo do valor percentual de cada fatia. Os indivíduos são selecionados por um valor aleatório que varia de 0 a 100, portanto aquele que detém a maior fatia na roleta terá maior probabilidade de ser selecionado, ver Figura 2.13.

2.3.6 Operador de Cruzamento

A forma de cruzamento aplicada utiliza apenas um ponto de corte, ou seja, os cromossomos são divididos em duas partes e recombinados dois a dois, o procedimento de cruzamento é realizado segundo uma probabilidade P_c que varia de $0.5 \leq \text{a} \leq 1.0$ como sugere (ANDRADE et al., 2008).

A maneira com que o ponto de corte é selecionado acontece de forma diferente da convencional. Para evitar um cruzamento que leve a produção de uma prole inviável, procedemos o cruzamento se os dois pais possuírem pelo menos um gene em comum (AHN; RAMAKRISHNA, 2002). Caso existam mais de um gene em comum, os seus respectivos *locus* são armazenados em uma lista e um número aleatório é utilizado para escolher dentre os possíveis *locus* da lista um ponto para a troca, ver Figura 2.14.

2.3.7 Operador de Mutação

O operador de mutação consiste na permuta de um gene C escolhido aleatoriamente entre 1 e $M-1$ por um de seus vizinhos, caso este esteja presente no cromossomo e esteja posicionado entre C e M, ver Figura 2.15. Esse procedimento evita que o operador mutação leve um cromossomo viável a tornar-se inviável. O percentual de mutação dos indivíduos é fixado a priori no algoritmo.

O algoritmo 3 descreve a solução proposta.

A Figura 2.16 mostra o fluxograma com os detalhes de cada etapa da execução do algoritmo proposto.

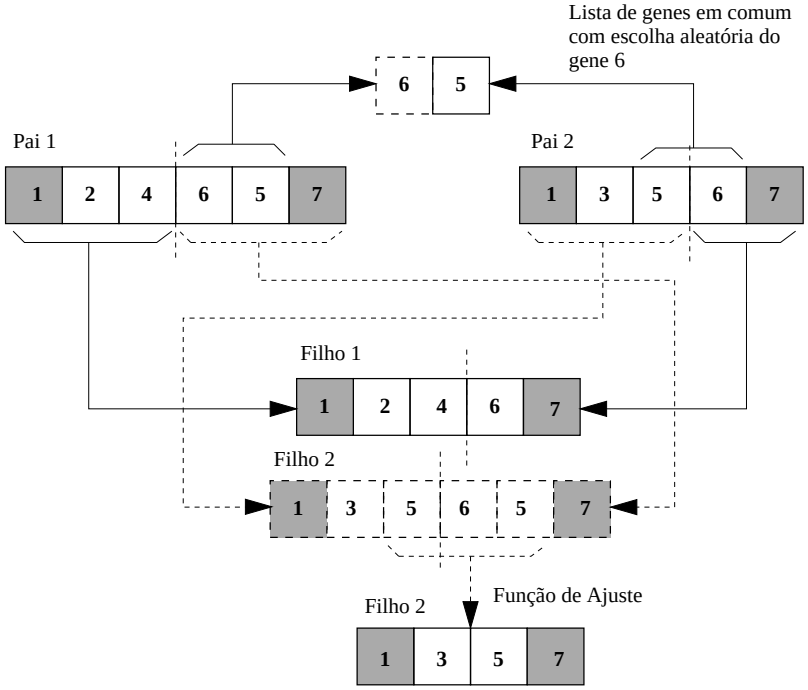


Figura 2.14: Exemplo do cruzamento entre indivíduos
Fonte: (Autor, 2012)

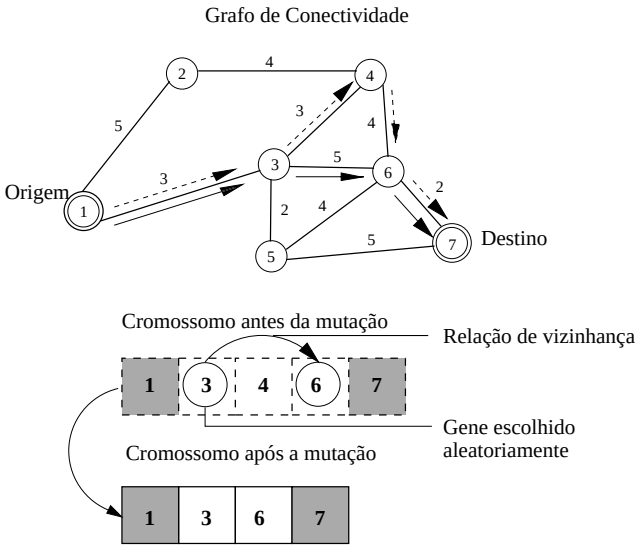


Figura 2.15: Exemplo de mutação em um indivíduo
Fonte: (Autor, 2012)

Entrada: Grafo de Conectividade, Nó Origem, Nó Destino
Saída: Melhor Indivíduo (Caminho Mínimo)
início
  para $i \leftarrow 1$ **até** N **faça**
    cromossomos[i][1] ← nó que contém o ponto Inicial;
    enquanto $j <$*qtd de nós do grafo e cromossomos*$[i] \neq$*Nó Destino* **faça**
      sorteia um vizinho de cromossomos[i];
      cromossomo[i][j] ← vizinho sorteado;
    fim
  fim
  para $i \leftarrow 1$ **até** N **faça**
    Utilizar a função de ajuste para remover nós repetidos e/ou laços do cromossomo[i];
    Calcular a aptidão do cromossomo[i];
  fim
  Remover cromossomos repetidos;
  enquanto *critério de parada não satisfeito* **faça**
    se *o numero de pais Aptos* < 2 **então**
      Completar população gerando indivíduos até o limite N;
      Utilizar a função de ajuste para remover nós repetidos e/ou laços;
    fim
    senão
      Calcular probabilidade de cruzamento **se** *probabilidade de cruzamento* $<=$ *taxa de cruzamento* **então**
        Calcular custo dos cromossomos utilizando função de valoração;
        Selecionar os Indivíduos mais aptos utilizando método da roleta;
        Verificar os pontos para realizar cruzamento;
        Cruzar aleatoriamente os indivíduos mais aptos até atingir N indivíduos;
        Calcular probabilidade de mutação **se** *probabilidade de mutação* $<=$ *taxa de mutação* **então**
          Aplicar aleatoriamente o operador de mutação;
        fim
      fim
      se *não atingiu N indivíduos* **então**
        Completar população gerando indivíduos aleatoriamente até o limite N;
      fim
    fim
  fim
fim

Algoritmo 3: Pseudocódigo do Algoritmo Proposto

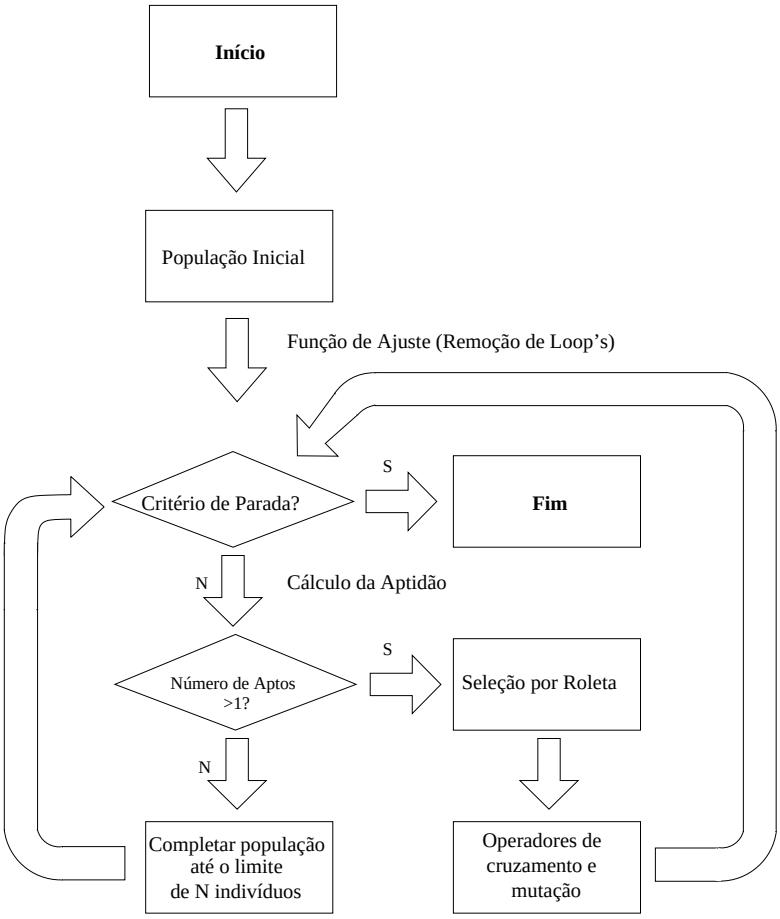


Figura 2.16: Fluxograma de execução do algoritmo proposto
Fonte: (Autor, 2012)

2.4 Arquivos utilizados pela aplicação

A aplicação foi desenvolvida na linguagem C e possui como dados de entrada, informações sobre a posição e quantidade dos obstáculos, posição de origem e destino do robô e dimensões do ambiente que são obtidas de um arquivo de texto chamado *"Dados.txt"*. Após o procedimento de geração dos *C-obstáculos* e obtenção do grafo de conectividade é aplicado o algoritmo de busca, gerando dois novos arquivos de saída: o *"Scilab.txt"* e o *"Trajetória.txt"*, como mostra a Figura 2.17.

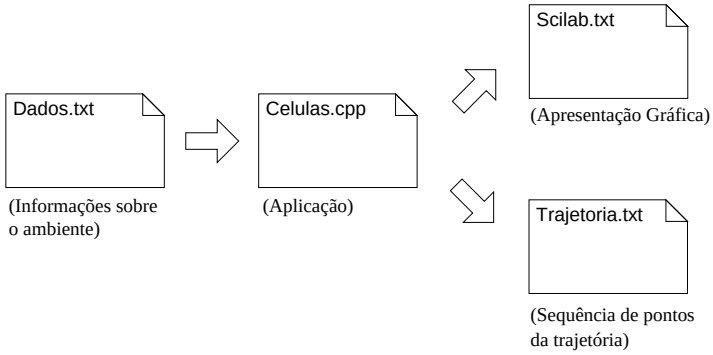


Figura 2.17: Arquivos utilizados pela aplicação
Fonte: (Autor, 2012)

Além dos parâmetros do ambiente e dos obstáculos, posições e orientações inicial e final do robô também são lidas do arquivo *"Dados.txt"* como na Figura 2.18. Os arquivos de saída são os pontos da trajetória dados pelo arquivo *"Trajetória.txt"*, ver Figura 2.19, que serão transmitidos ao robô pelo enlace de rádio. As informações que serão utilizadas para visualização gráfica da trajetória executada pelo robô são obtidas pelo processamento do arquivo *"Scilab.txt"*, mostrado na Figura 2.20.

Na Figura 2.18 podem ser observados os parâmetros de entrada como a quantidade de obstáculos e logo em seguida para cada um dos obstáculos são dadas as coordenadas de cada vértice. Na sequência são dadas as dimensões do ambiente e por fim as informações de posição e orientação inicial e final do robô.

A Figura 2.21 mostra o resultado do processamento do arquivo "Scilab.txt" gerando um gráfico que permite a visualização da trajetória obtida pela aplicação do método de decomposição em células juntamente com o algoritmo de busca.

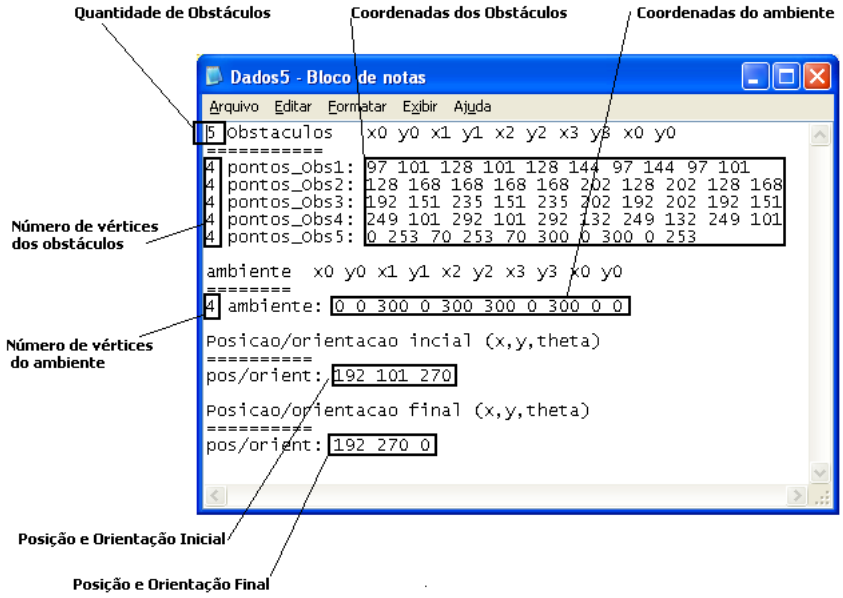


Figura 2.18: Detalhe do arquivo de entrada "Dados.txt"
Fonte: (Autor, 2012)

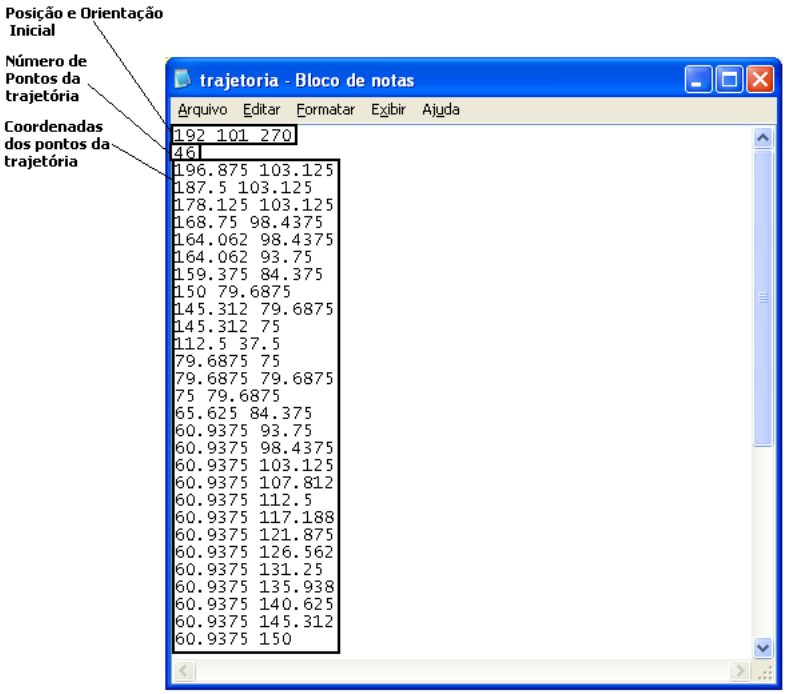


Figura 2.19: Detalhe do arquivo de saída "Trajetoria.txt"
Fonte: (Autor, 2012)

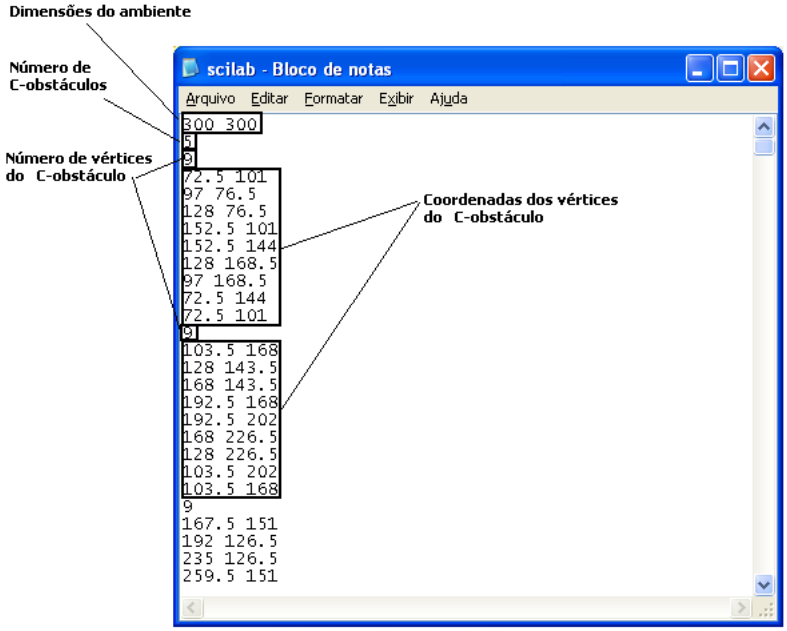


Figura 2.20: Detalhe do arquivo de saída "Scilab.txt"
Fonte: (Autor, 2012)

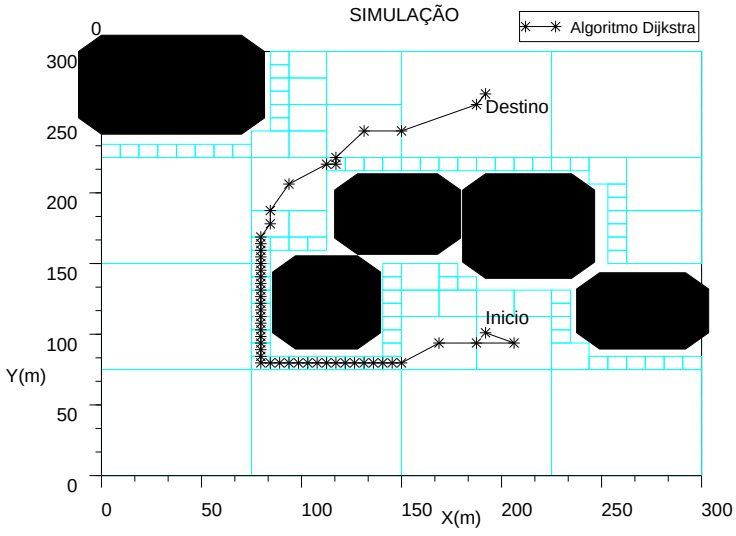


Figura 2.21: Visualização gráfica da trajetória
Fonte: (Autor, 2012)

2.5 Plataforma Robótica utilizada nos testes

O sistema robótico utilizado nos experimentos é composto por uma placa responsável pela transmissão das coordenadas através de um enlace de rádio e pelo robô móvel que recebe as coordenadas, as decodifica e as executa. A placa transmissora é conectada a um *PC* por meio de uma *interface* USB/Serial. No *PC*, o aplicativo de geração de trajetórias é o responsável pelo envio das coordenadas à placa. Uma visão geral do sistema pode ser observado na Figura 2.22.

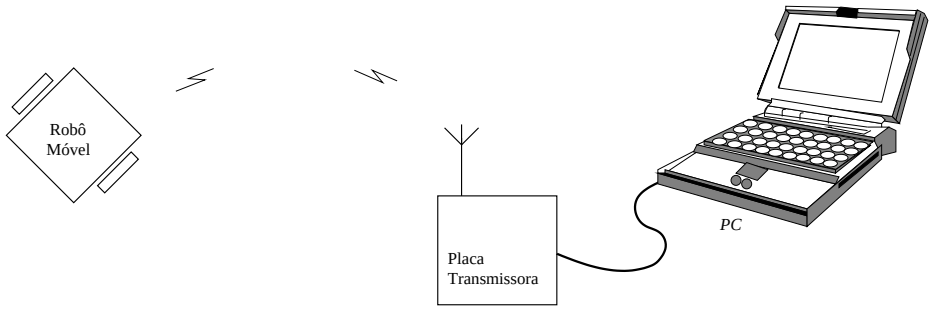
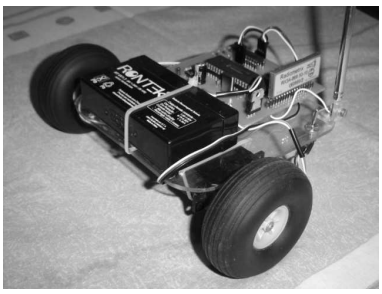
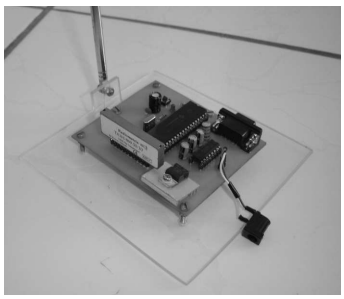


Figura 2.22: Visão geral dos componentes do sistema robótico utilizado nos experimentos
Fonte: (Autor, 2012)

A Figura a 2.23 (a), mostra o robô móvel utilizado nos experimentos e a Figura 2.23 (b), o detalhe da placa transmissora.



a)Robô móvel



b)Placa transmissora

Figura 2.23: Robô e placa de transmissão utilizados nos experimentos

Fonte: (Autor, 2012)

Capítulo 3

Resultados e Discussões

Neste Capítulo serão apresentados os resultados de testes aplicando o algoritmo proposto. Serão apresentadas comparações com o algoritmo de Dijkstra que foi utilizado como referência na obtenção do caminho ótimo. As simulações foram implementadas na linguagem C em um processador Intel® Pentium® Dual Core (1.46 GHz de clock) 2GB de memória RAM. Em todas as simulações aplicou-se para seleção o método da roleta, descrito em detalhes na Seção 2.3.5 do Capítulo 2 e uma probabilidade de mutação fixada em 5% e probabilidade de cruzamento fixada em 80%.

Esses valores foram obtidos após resultados de simulações que mostraram que um aumento no número de mutações não conduzem a perda diversidade da população mas também não representaram ganho significativo relativos a convergência do algoritmo. A probabilidade de cruzamento em torno de 80% garante uma inserção de novos indivíduos a cada iteração. Cada ponto de cada gráfico foi obtido para um total de 1000 amostras sendo representado por uma média e desvio padrão.

A convergência acontece quando não ocorre alteração no melhor indivíduo de uma população durante três gerações. Para as primeiras simulações mostradas nas Figuras 3.2 e 3.4, aplicou-se, como forma de geração dos indivíduos, o método totalmente aleatório. Após fixar o gene com o ponto inicial, procede-se o sorteio do gene seguinte dentre todos os nós do grafo. As Figuras 3.1 e 3.3 ilustram os dois modelos aplicados às simulações 1 e 2.

Esse procedimento resulta em um número baixo de cruzamentos e o custo computacional é muito dependente do tamanho da população e do número de células que formam o caminho. O critério aleatório para a inicialização não apresentou bons resultados para grafos de conectividade com um número de nós maior que 12 ou quando a quantidade de nós do caminho era superior a 5. A Figura 3.2 apresenta um comparativo da aplicação do algoritmo proposto com o algoritmo de Dijkstra observando a evolução do tempo de processamento (média e desvio padrão) em

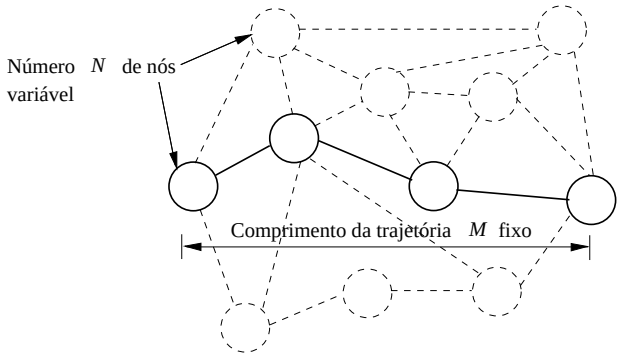


Figura 3.1: Representação gráfica do modelo aplicado ao experimento 1, com variação do número de nós do grafo de conectividade N, mantendo fixo o comprimento M da trajetória
Fonte: (Autor, 2012)

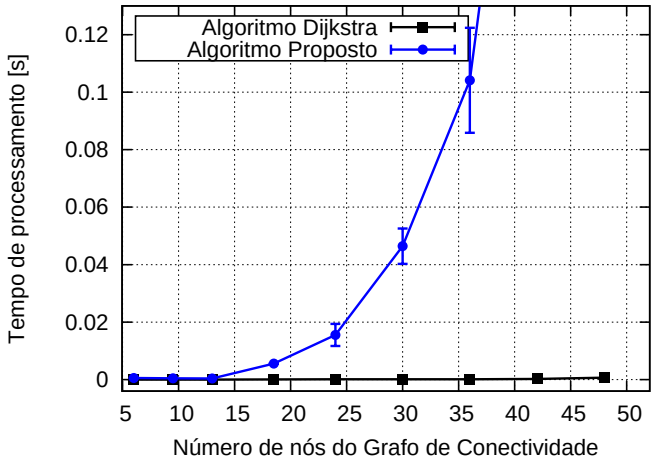


Figura 3.2: Comparativo para o tempo de processamento versus número de nós do grafo de conectividade com inicialização dos cromossomos utilizando critério aleatório
Fonte: (Autor, 2012)

relação a quantidade de nós da trajetória. Na Figura 3.4 pode-se observar outro comparativo, agora considerando o tempo de processamento em relação ao número de nós do caminho. Para os experimentos das Figuras 3.2 e 3.4 utilizou-se uma população de 200 indivíduos, probabilidade de cruzamento de 80%, probabilidade de mutação de 5% e a média foi obtida para um total de 1000 amostras para cada ponto dos gráficos.

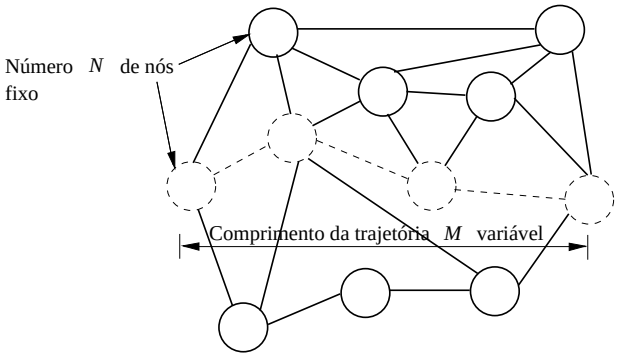


Figura 3.3: Representação gráfica do modelo aplicado ao experimento 2, com variação do comprimento M da trajetória, mantendo fixo o número de nós do grafo de conectividade N

Fonte: (Autor, 2012)

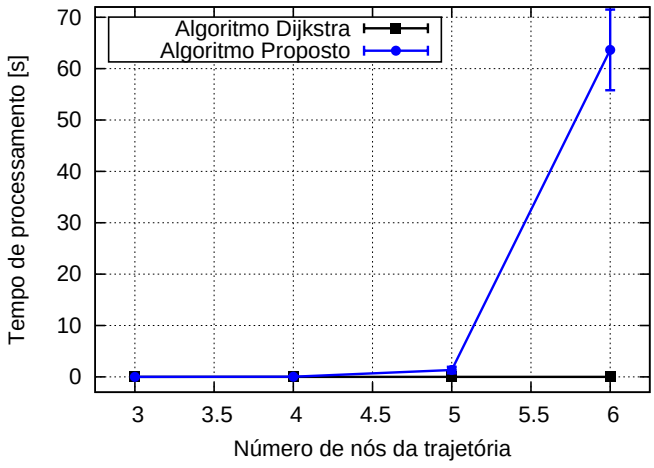


Figura 3.4: Comparativo para o tempo de processamento versus número de nós da trajetória com inicialização dos cromossomos utilizando critério aleatório

Fonte: (Autor, 2012)

Com base nesses resultados percebeu-se a inviabilidade da inicialização dos cromossomos utilizando critério aleatório. Para resolver este problema, passamos

a utilizar um critério heurístico no procedimento de geração de indivíduos. Neste caso, os cromossomos tem o primeiro gene do cromossomo fixado com o nó inicial e os nós subsequentes são sorteados dentre os vizinhos do nó corrente. Desta forma, o número de indivíduos viáveis da população aumentou consideravelmente, permitindo uma convergência mais rápida.

Na Figura 3.5 pode-se observar o resultado com valor médio após 1000 amostras para cada ponto após a utilização da inicialização dos indivíduos pelo sorteio aleatório dos vizinhos de cada gene, obtendo-se um tempo de convergência menor do que o algoritmo de Dijkstra, com erro máximo permitido em relação ao resultado ótimo de até 20%.

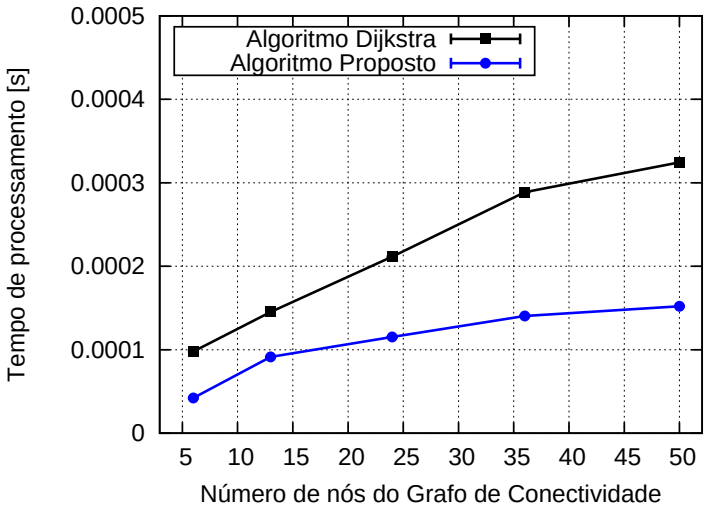


Figura 3.5: Comparativo para o tempo de processamento versus número de nós do grafo de conectividade com inicialização dos cromossomos utilizando critério de sorteio dos vizinhos
Fonte: (Autor, 2012)

As simulações a seguir, mostram a influência do tamanho da população na qualidade da solução obtida para uma probabilidade de cruzamento de 80% e probabilidade de mutação de 5%. Quanto maior o número de indivíduos mais o resultado do algoritmo se aproxima da solução ótima. Na Figura 3.6 pode-se observar o caminho alternativo obtido pelo algoritmo proposto com custo médio de 492,55 cm para uma população de 20 indivíduos, 26,58% maior em relação ao resultado ótimo dado pelo algoritmo Dijkstra de 389,12 cm. Na Figura 3.7, observa-se uma pequena redução no custo da solução obtida pelo algoritmo proposto que passa a ter média de 459,50 cm para uma população de 50 indivíduos, 18,08% maior que o ótimo.

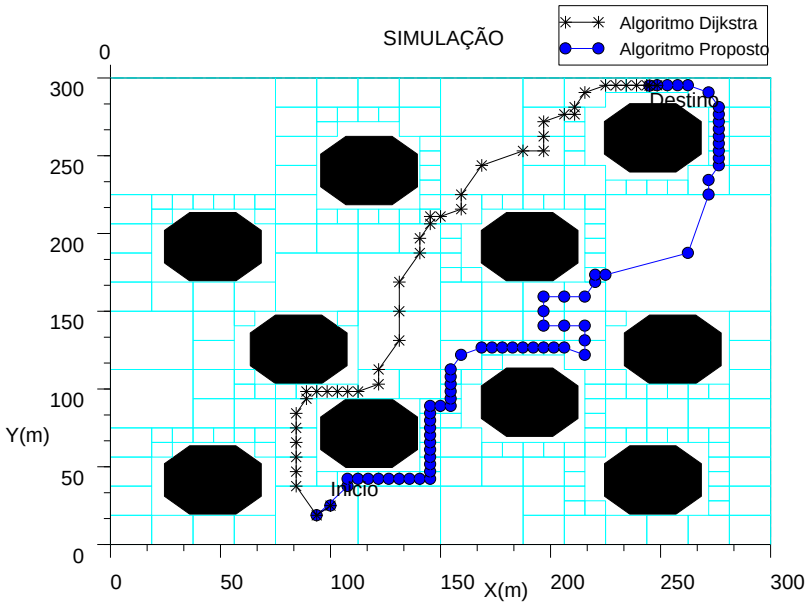


Figura 3.6: Trajetória obtida pelo algoritmo proposto com uma população de 20 indivíduos, 26,58% maior em relação ao resultado do algoritmo Dijkstra
Fonte: (Autor, 2012)

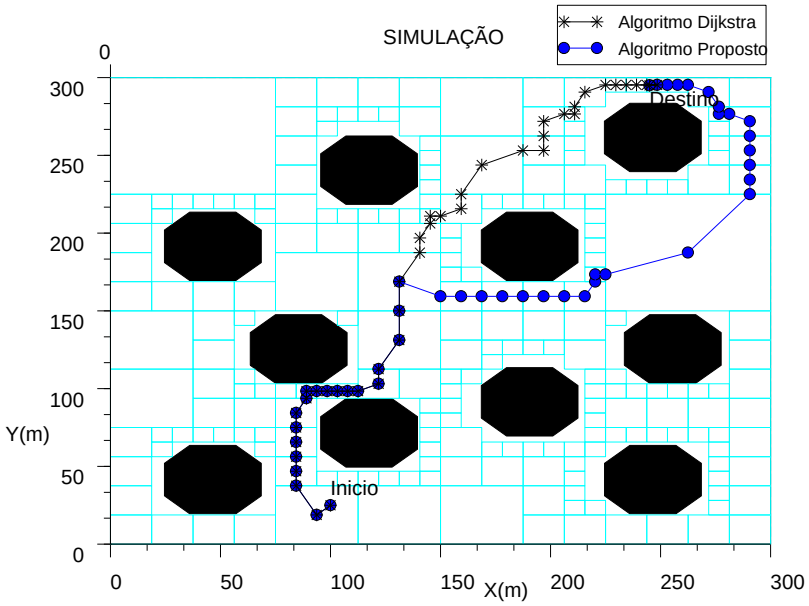


Figura 3.7: Trajetória obtida pelo algoritmo proposto com uma população de 50 indivíduos, 18,08 % maior em relação ao resultado do algoritmo Dijkstra
Fonte: (Autor, 2012)

A Figura 3.8 mostra uma aproximação mais significativa do trajeto ótimo com custo médio de 430,20 cm, 10,55% maior que o ótimo para uma população de 150 indivíduos. Finalmente na Figura 3.9 pode-se observar praticamente uma coincidência nas trajetórias obtidas pelos algoritmos, agora com média de 398,49 cm para uma população de 200 indivíduos que corresponde a uma variação de 2,40% maior que o ótimo. Os valores médios e seus desvios foram obtidos para 1000 iterações em cada situação. O resumo dos experimentos pode ser observado na tabela 3.1.

Tabela 3.1: Resultados obtidos aplicando-se o método de inicialização heurístico

Tam. População	Dist. Média(cm)	Des. Padrão(cm)	Dist. Ótima(cm)	Erro(%)
20	492,55	18,99	389,12	26,58
50	459,50	16,89	389,12	18,08
150	430,20	15,26	389,12	10,55
200	398,49	14,85	389,12	2,40

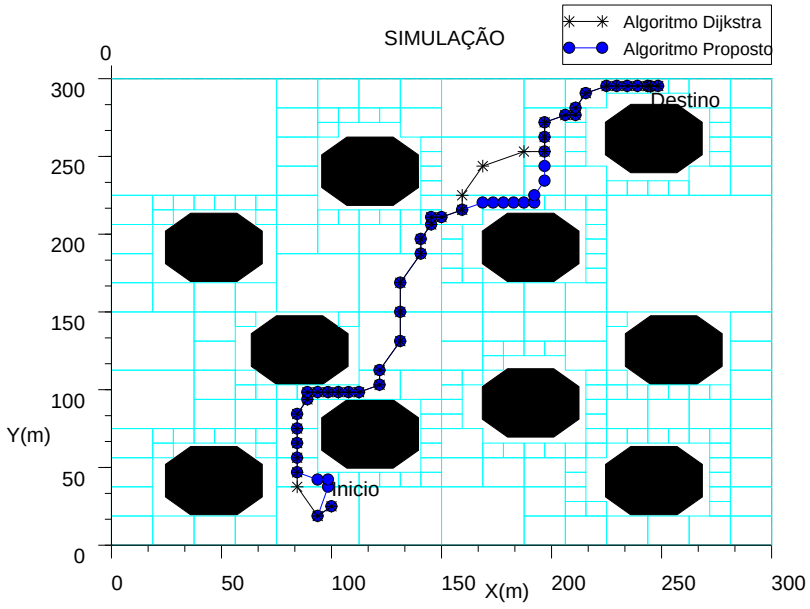


Figura 3.8: Trajetória obtida pelo algoritmo proposto com uma população de 150 indivíduos, 10,55% maior em relação ao resultado do algoritmo Dijkstra
Fonte: (Autor, 2012)

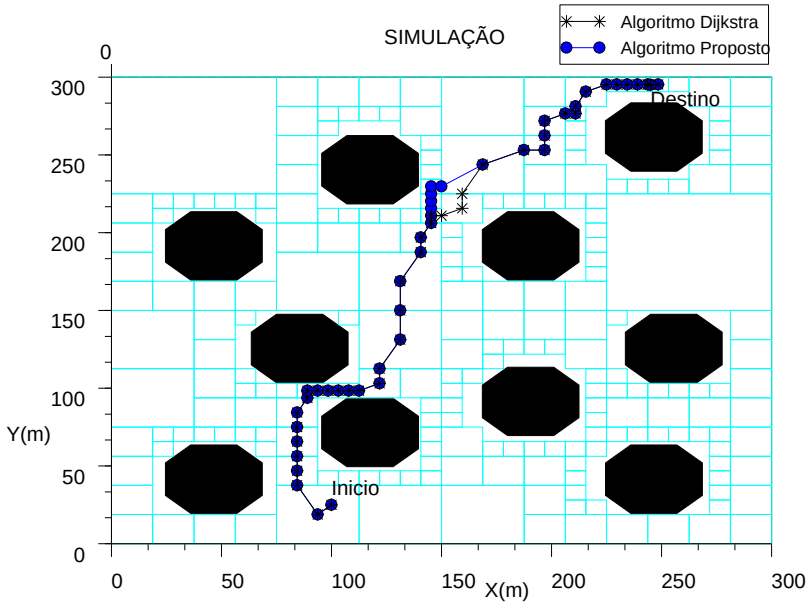


Figura 3.9: Trajetória obtida pelo algoritmo proposto com uma população de 200 indivíduos, 2,40 % maior em relação ao resultado do algoritmo Dijkstra
Fonte: (Autor, 2012)

O gráfico da Figura 3.10 mostra a comparação para o tempo de processamento em relação ao número de indivíduos da população para as configurações das Figuras 3.6 a 3.9. Pode-se observar que o custo computacional cresce com o aumento do número de indivíduos, consequentemente aumentando a qualidade das soluções obtidas pelos mesmos experimentos. Entretanto, deve-se fazer uma análise para verificar quão grande deve ser a população utilizada pelo algoritmo. Através dessa análise, pode-se evitar o aumento do custo computacional para obtenção de uma solução cada vez melhor ou que satisfaça os requisitos da aplicação sem necessariamente ser ótima.

A simulação a seguir foi realizada com o objetivo de verificar o tempo de processamento necessário à execução de cada uma das etapas do algoritmo, com o intuito de indicar qual demanda mais processamento, possibilitando um diagnóstico que possibilite uma redução no custo computacional. Nesta simulação foram comparados os resultados na utilização dos dois métodos de inicialização para cromossomos implementados, o método de inicialização aleatório, descrito em detalhes nos experimentos 1 e 2, Figuras 3.2 e 3.4 e o método com a heurística da inicialização dos genes através da vizinhança do gene atual, descrito em detalhes nos experimentos da Figura 3.5 a 3.8.

Para fazer uma comparação entre o desempenho nas duas situações foi obtida

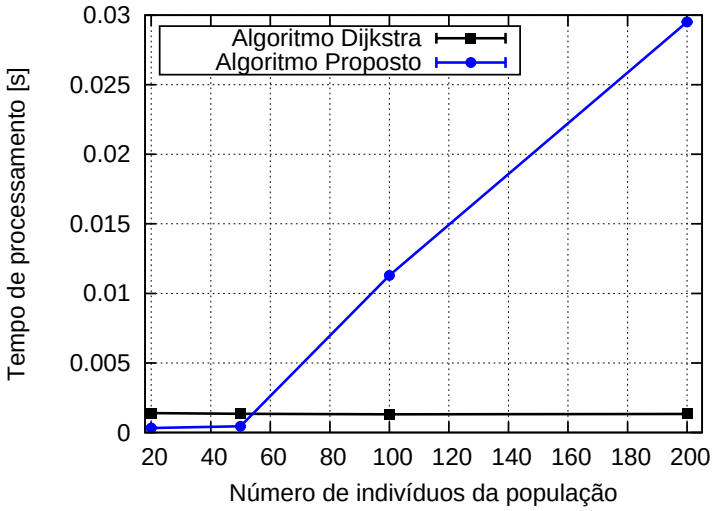


Figura 3.10: Comparativo para o tempo de processamento versus número de indivíduos da população para as configurações das Figuras 3.6 a 3.9
Fonte: (Autor, 2012)

uma média para cada etapa de um total de 1000 iterações utilizando cada método, considerando apenas a primeira geração em cada iteração. Foram utilizadas as probabilidades de cruzamento e mutação de 80% e 5% respectivamente e as mesmas configurações de ambiente. Neste experimento pode-se ainda verificar o número de iterações necessárias para convergência com o uso de cada método.

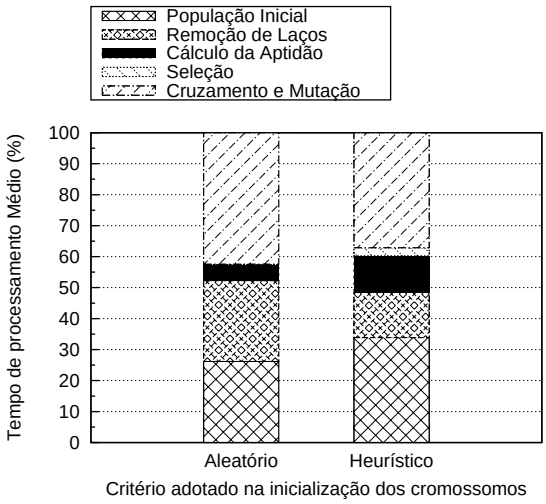


Figura 3.11: Tempo de processamento médio em cada etapa do algoritmo proposto utilizando os critérios: Aleatório e Heurístico para inicialização dos cromossomos
Fonte: (Autor, 2012)

A Figura 3.11 mostra o resultado da simulação, aqui, pode-se observar o aumento no tempo de processamento destinado a etapa de geração da população inicial utilizando o critério heurístico. Esse fato ocorre em função da pesquisa e sorteio entre os vizinhos de cada gene. Como o número de cromossomos aptos é maior para a situação heurística, consequentemente ocorre um aumento no percentual referente ao cálculo da aptidão dos indivíduos. Ainda na primeira geração pode-se observar um número diferente de zero de cromossomos aptos obtidos pela presença do percentual referente à etapa de seleção. A etapa de cruzamento e mutação envolve também a geração de uma nova população quando o número de descendentes é menor que o total de indivíduos estabelecidos para a população. Em virtude deste fato, o percentual referente a esta etapa não reflete fielmente o número de cruzamentos realizados, mas sim o tempo desprendido para inserção dos novos indivíduos na população e a geração de outros para complementação do tamanho N. O ganho em número de cromossomos aptos pelo critério heurístico pode ser visto pelo número de gerações necessárias a convergência como mostrado na tabela 3.2 que apresenta os valores brutos da simulação.

Tabela 3.2: Valores do experimento para avaliação do tempo de processamento em cada etapa do algoritmo

Etapa	Critério de Inicialização	
	aleatório	heurístico
	tempo(%)	tempo(%)
Pop. Inicial	26,18	33,88
Rem. Laços	26,10	14,56
Cál. Aptidão	5,29	11,78
Seleção	0,00	2,63
Cruz. e Mutação	42,43	37,15
Total	100,00	100,00
Total de iterações para convergência	723	3

Para comprovação da aplicação prática do algoritmo proposto foram realizados testes experimentais em um ambiente estruturado real. No primeiro experimento pode-se observar a trajetória obtida pela aplicação do algoritmo proposto para um ambiente com três obstáculos dispostos como ilustra a Figura 3.12. Nesta configuração, em virtude de não haver células livres entre os *C-Obstáculos* o caminho encontrado entre os nós de início e destino corresponde ao contornar dos obstáculos. A Figura 3.13 mostra a execução da trajetória da Figura 3.12 pelo robô móvel.

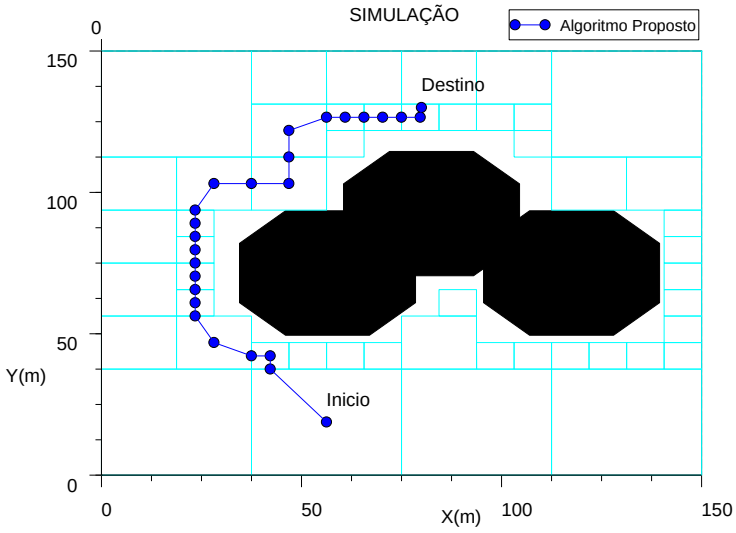


Figura 3.12: Configuração de ambiente com três obstáculos
Fonte: (Autor, 2012)

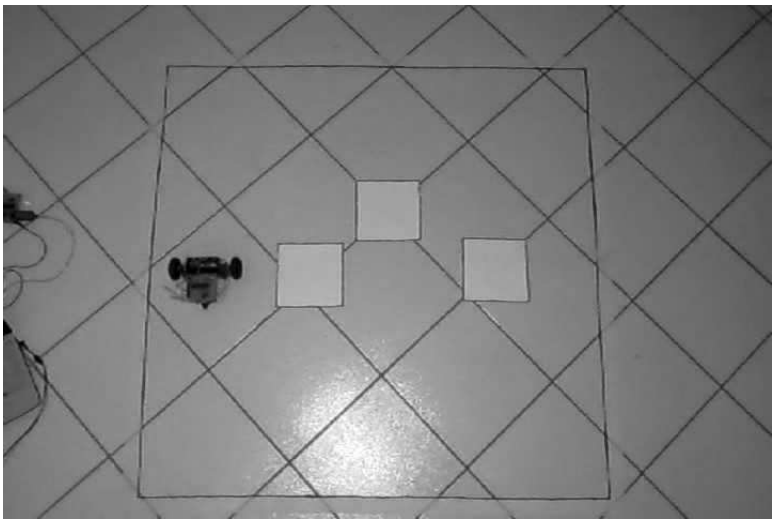


Figura 3.13: Robô móvel executando trajetória da Figura 3.12
Fonte: (Autor, 2012)

No segundo experimento removemos o obstáculo central para observar o comportamento do algoritmo e a nova trajetória gerada, agora por entre os *C-Obstáculos* centrais como mostra a Figura 3.14. Na Figura 3.15 a execução desta trajetória pelo robô móvel.

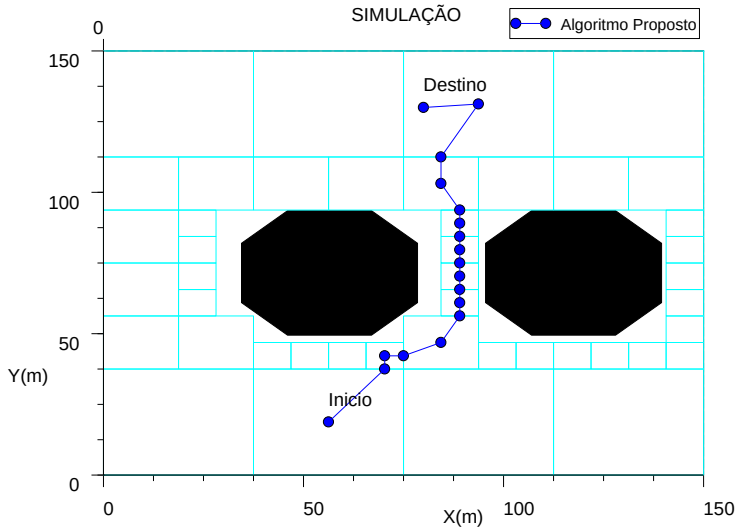


Figura 3.14: Configuração de ambiente com dois obstáculos
Fonte: (Autor, 2012)

Figura 3.15: Robô móvel executando trajetória da Figura 3.14
Fonte: (Autor, 2012)

Capítulo 4

Conclusões e Trabalhos futuros

O estudo conjunto de técnicas inteligentes, em especial os algoritmos genéticos e algoritmos de busca em grafos, possibilitou o desenvolvimento e aplicação de um algoritmo evolucionário para uso em planejamento de caminhos para robôs móveis com resultados bastante satisfatórios com relação a algoritmos de busca tradicionais como é o caso do algoritmo de Dijkstra, utilizado como medida padrão nas simulações realizadas por obter sempre resultados ótimos.

Os experimentos realizados mostraram que o algoritmo proposto consegue tempos de convergência menores do que o algoritmo de Dijkstra gerando resultados bastante próximos do resultado ótimo (caminho mais curto) e que o número de indivíduos da população influencia diretamente na qualidade deste resultado (tamanho da trajetória), cabendo uma análise relacionando tempo de processamento com qualidade da solução para definição mais adequada do tamanho da população. Um aspecto importante que deve ser considerado é que na pesquisa bibliográfica realizada não foram encontrados registros de trabalhos utilizando algoritmos genéticos aplicados como ferramenta de busca em grafos obtidos a partir de técnicas de planejamento de caminhos.

Esses grafos possuem uma característica particular que é a esparcidade (número de vértices próximo ao de arestas) como consequência da presença de obstáculos no ambiente. Essa característica pode levar algoritmos de critério guloso como o Dijkstra a buscas desnecessárias, o que é minimizado pela característica estocástica do algoritmo genético. Os trabalhos que deram suporte ao desenvolvimento da proposta constituiram-se por algoritmos genéticos aplicados a problemas de roteamento em interconexão de sistemas de redes de computadores que contribuíram com idéias que auxiliaram a construir principalmente etapas do algoritmo como codificação e cruzamento e seleção. No algoritmo proposto o operador de mutação apresentado difere das implementações convencionais pela realização de uma verificação e busca por pontos (*locus*) de troca (permuta) que não levam o indivíduo

a inviabilidade como ocorre com muita frequência no uso deste operador para problemas de busca em grafos.

Como trabalhos futuros tem-se como idéia inicial a otimização das estruturas de dados utilizadas na implementação prática do algoritmo e melhoria nas técnicas de manipulação destas nos procedimentos de geração de indivíduos e cruzamento que correspondem a cerca de 70% do tempo total gasto para executar o algoritmo. Pretende-se também, utilizar algoritmos genéticos adaptativos, para buscar uma redução do tempo computacional nas etapas de geração de indivíduos e cruzamento.

Referências Bibliográficas

AHN, C. W.; RAMAKRISHNA, R. S. A. A genetic algorithm for shortest path routingproblem and the sizing of populations. In: *Proceedings of IEEE Transactions on Evolutionary Computation*. [S.l.: s.n.], 2002.

ALMEIDA, C. O.; ZIVIANI, N. *Notas de Aula - Projeto de Algoritmos Cap.7 Algoritmos em Grafos*. 2004.

ALSINA, P. J. *Nota de Aula n.6 - Disciplina: Sistemas Robóticos Autonomos*. 2001.

ANDRADE, A. et al. Analysis of selection and crossover methods used by genetic algorithm-based heuristic to solve the lsp allocation problem in mpls networks under capacity constraints. In: *Proceedings of International Conference on Engineering Optimization*. [S.l.: s.n.], 2008.

BARROS, E. A. R.; PAMBOUKIAN, S. V. D.; ZAMBONI, L. C. Algoritmo de dijkstra: Apoio didático e multidiciplinar na implementação, simulação e utilização computacional. *International Conference on Engineering and Computer Education - ICECE*, March 2007.

BORENSTEIN, J. et al. Mobile robot positioning and sensors and techniques. *Robotic Systems, Special Issue on Mobile Robots*, v. 14, n. 4, p. 231–249, April 1997.

BRUCE, J. R.; VELOSO, M. Real-time randomized path planning for robot navigation. In: *Proceedings of IRDS'2002*. [S.l.: s.n.], 2002. p. 2383–2388.

CAKIR, M.; BUTUN, E.; KAYMAN, Y. Effects of genetic algorithm parameters on trajectory planning for 6-dof industrial robots. *Industrial Robot: An International Journal*, v. 33, p. 205–215, 2006.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. *Distributed Systems: Concepts and Design*. 4. ed. [S.l.]: Addison-Wesley, 2005. ISBN 0321263545.

DIESTEL, R. *Graph Theory*. 4. ed. New York USA: Springer-Verlag, Heidelberg, 2010. ISBN 978-3-642-14278-9.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1. ed. [S.l.]: Addison-Wesley, 1989. ISBN 978-0201157673.

INAGAKI, J.; HASEYAMA, M.; KITAJIMA, H. A genetic algorithm for determining multiple routes and its applications. In: *Proceedings of IEEE Int. Symp. Circuits and Systems*. [S.l.: s.n.], 1999. p. 137–140.

KANG, J. M. *Voronoi Diagram - Encyclopedia of GIS*. 1. ed. Universidade de Minesota, USA: Springer, 2008. 1232-1235 p.

LATOMBE, J. C. *Robot motion planning*. 2. ed. Massachusetts USA: Kluwer Academic Publishers, 1991. ISBN 0-7923-9129-2.

MUNEMOTO, M.; TAKAI, Y.; SATO, Y. A migration scheme for the genetic adaptive routing algorithm. In: *Proceedings of IEEE Int. Conf. Systems, Man, and Cybernetics*. [S.l.: s.n.], 1998. p. 2774–2779.

NAGIB, G.; ALI, W. G. Network routing protocol using genetic algorithms. *International Journal of Eletrical & Computer Sciences IJECS-IJENS*, v. 10, n. 2, March 2010.

OTTONI, G. d. L.; LAGES, W. F. Planejamento de trajetórias para robôs móveis em ambientes desconhecidos. In: *Proceedings of CBA'2000*. [S.l.: s.n.], 2000. p. 2239–2244.

PACHECO, M. A. *Algoritmos Genéticos: Princípios e Aplicações*. [S.l.], 1999.

PEREZ, A.; ROSELL, J. A roadmap to robot motion planning software development. *Computer Applications in Engineering Education*, v. 18, n. 4, p. 651–660, December 2010.

SCHRAFT, R. Mechatronics and robotics for services applications. *IEEE Robotics and Automation Magazine*, December 1994.

SICILIANO, A. d. V. *Determinação de Trajetória ótima em Navegação Robótica Móvel, Utilizando Algoritmo Genético.* Dissertação (Mestrado) — Universidade Federal do Rio de Janeiro, COPPE/UFRJ, Rio de Janeiro, RJ, Julho 2006.

SIVARAJ, R.; EVERETT, T. R. A review of selection methods in genetic algorithm. *International Journal of Engineering Science and Technology - IJEST*, v. 3, n. 5, May 2011.

SLEUMER, N. H.; TSCHICHOLD-GüRMANN, N. *Exact cell decomposition of arrangements used for path planning in robotics.* [S.l.], 1999.

SOUZA, S. C. B. *Planejamento de Trajetória para um robô móvel com duas rodas utilizando um algoritmo A-Estrela Modificado.* Dissertação (Mestrado) — Universidade Federal do Rio de Janeiro, COPPE/UFRJ, Rio de Janeiro, RJ, Dezembro 2008.

SWAMINATHAN, G. *Robot Motion Planning.* Canadá, February 2006. School of Engineering Science Simon Fraser University, Canada.

ZENG, C.; ZHANG, Q.; WEI, X. Robotic global path-planning based modified genetic algorithm and a* algorithm. In: *Proceedings of Third International Conference on Measuring Technology and Mechatronics Automation.* [S.l.: s.n.], 2011. p. 167–170.

Apêndice A

Sistema robótico utilizado nos experimentos

O protótipo construído possui acionamento diferencial realizado por dois servomotores com *encoders* acoplados, conectados a uma placa eletrônica formada essencialmente por um microcontrolador, um drive para acionamento dos motores e um módulo receptor de rádio. Este último tem por função receber e decodificar as informações que são enviadas pelo módulo transmissor. Na placa de transmissão estão mais um microcontrolador e o módulo transmissor de rádio. A placa transmissora recebe os comandos pela porta de comunicação serial do computador e os envia para o robô pelo enlace de rádio. A figura A.1 mostra os principais componentes do robô e a figura A.2 mostra as principais vistas de seu projeto.

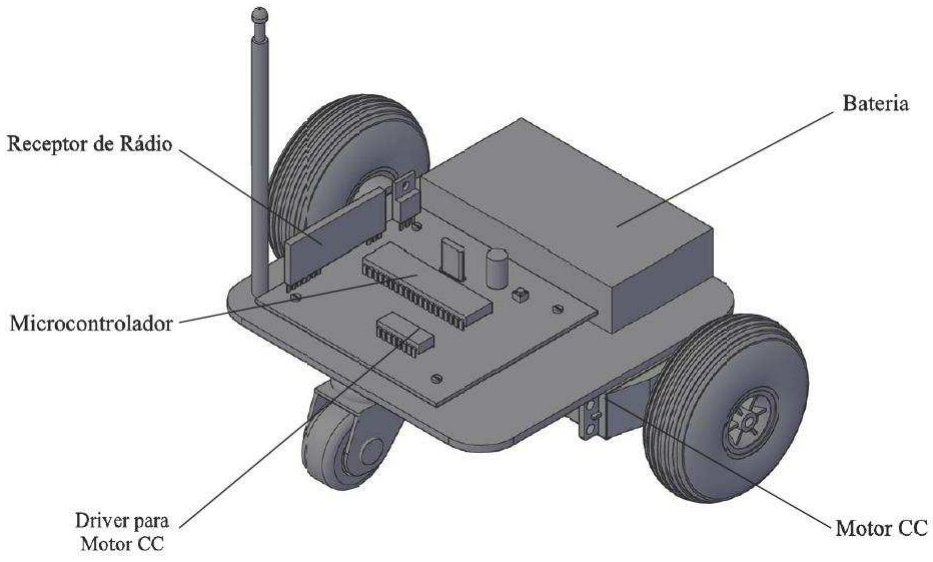
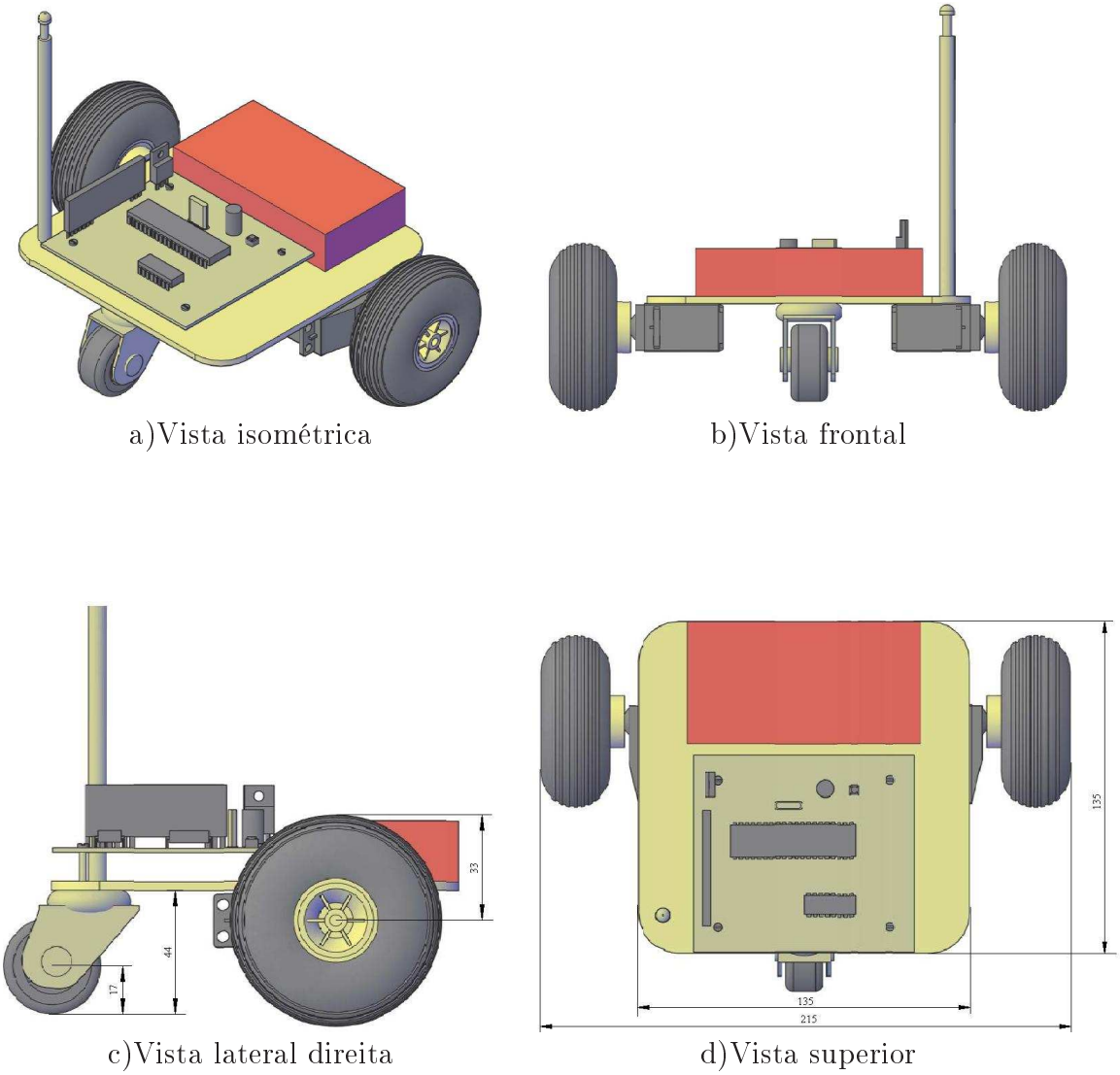


Figura A.1: Principais componentes do robô
Fonte: (Autor, 2012)



a)Vista isométrica b)Vista frontal

c)Vista lateral direita d)Vista superior

Figura A.2: Vistas principais do robô (cotas em mm)
Fonte: (Autor, 2012)

Apêndice B

Modelagem cinemática do Robô

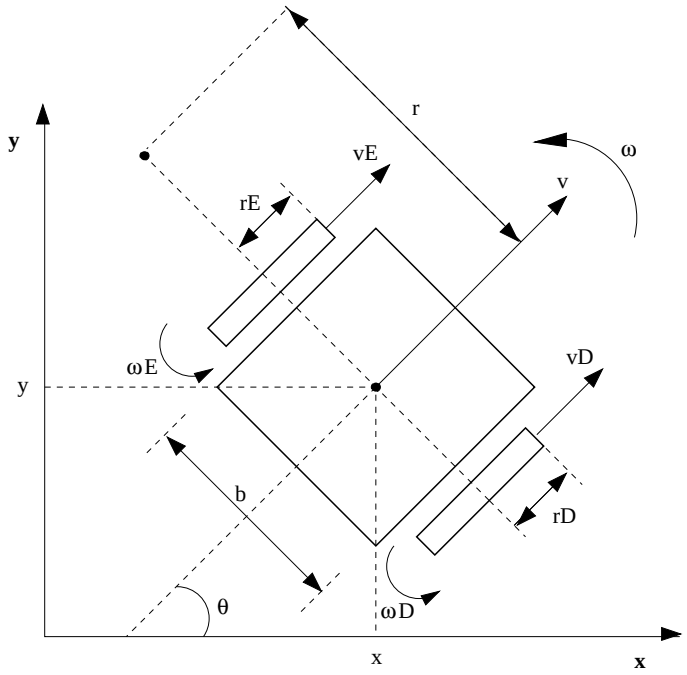


Figura B.1: Parâmetros cinemáticos do Robô
Fonte: (ALSINA, 2001)

Para a figura B.1 tem-se que:

(x, y) é a posição do referencial fixo no robô em relação ao referencial fixo no espaço de trabalho, θ é o ângulo de orientação do robô em relação ao referencial fixo no espaço de trabalho, b é Comprimento do eixo, r é o raio de giro do robô, $rD(rE)$ o raio da roda direita (esquerda), ω a velocidade angular do robô, $\omega D(\omega E)$ a velocidade angular da roda direita (esquerda), v a velocidade linear do robô ($v = \omega.r$) e $vD(vE)$ a velocidade linear da borda da roda direita (esquerda).

Para movimentos infinitesimais tem-se:

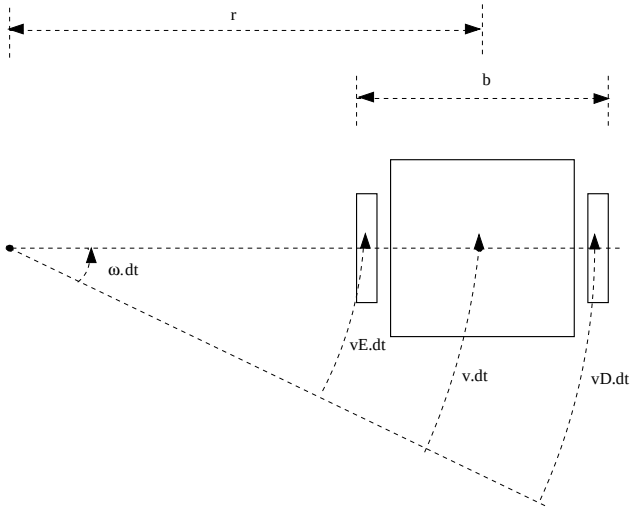


Figura B.2: Velocidades para movimentos infinitesimais
Fonte: (ALSINA, 2001)

$$\begin{cases} vD.dt = \omega(r + b/2)dt \\ vE.dt = \omega(r - b/2)dt \end{cases} \Longrightarrow \begin{cases} vD + vE = \omega D.rD + \omega E.rE = 2.\omega.r = 2.v \\ vD - vE = \omega D.rD - \omega E.rE = \omega.b \end{cases} \tag{B.1}$$

Escrevendo na forma vetorial:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} (rD/2) & (rE/2) \\ (rD/b) & -(rE/b) \end{bmatrix} . \begin{bmatrix} \omega D \\ \omega E \end{bmatrix} \Longrightarrow \mathbf{V} = {}^{\mathbf{V}}\,\mathbf{T_W}.\mathbf{W} \tag{B.2}$$

Onde:

$$\mathbf{V} = [\ v \quad \omega\ ]^T\ ,\ \mathbf{W} = [\ \omega D \quad \omega E\ ]^T \text{ e } {}^{\mathbf{V}}\mathbf{T_W} = \begin{bmatrix} (rD/2) & (rE/2) \\ (rD/b) & -(rE/b) \end{bmatrix}$$

Para deslocamentos infinitesimais tem-se:

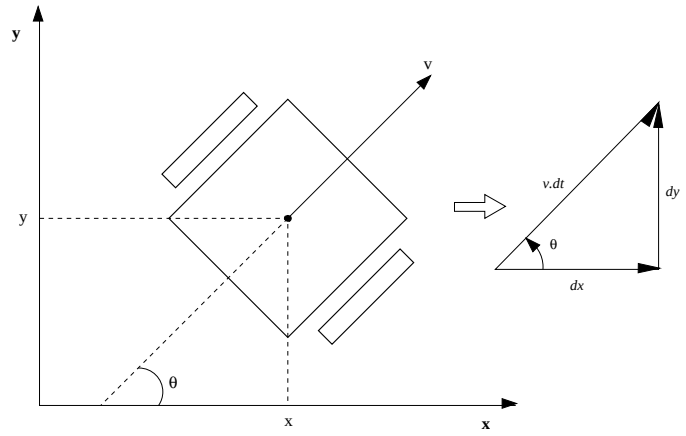


Figura B.3: Deslocamentos para movimentos infinitesimais
Fonte: (ALSINA, 2001)

Aplicando relações trigonométricas ao triângulo retângulo da figura B.3 obtem-se:

$$\begin{cases} dx = v.dt.cos\theta \\ dy = v.dt.sen\theta \\ d\theta = \omega.dt \end{cases} \Longrightarrow \begin{cases} dx/dt = \dot{x} = v.cos\theta \\ dy/dt = \dot{y} = v.sen\theta \\ d\omega/dt = \dot{\omega} = w \end{cases} \tag{B.3}$$

Ainda da figura B.3 pode-se obter:

$$\begin{cases} (dy/dx) = tan(\theta) = sen\theta/cos\theta \\ v.dt = dx.cos\theta + dy.sen\theta \end{cases} \Longrightarrow \quad dy.cos\theta - dx.sen\theta = 0$$

Representando na forma matricial tem-se:

$$\begin{bmatrix} cos\theta & sen\theta \\ -sen\theta & cos\theta \end{bmatrix} . \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} v \\ 0 \end{bmatrix} \tag{B.4}$$

Definindo o vetor de variáveis de configuração $\mathbf{q} = [\ x \quad y \quad \theta\ ]^T$, obtem-se o modelo cinemático:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos\theta & 0 \\ sen\theta & 0 \\ 0 & 1 \end{bmatrix} . \begin{bmatrix} v \\ \omega \end{bmatrix} onde, \begin{bmatrix} cos\theta & 0 \\ sen\theta & 0 \\ 0 & 1 \end{bmatrix} = {}^{\mathbf{q}}\mathbf{T}_{\mathbf{v}} \tag{B.5}$$

Substituindo (B.2) em (B.5) tem-se:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos\theta & 0 \\ sen\theta & 0 \\ 0 & 1 \end{bmatrix} . \begin{bmatrix} (rD/2) & (rE/2) \\ (rD/b) & -(rE/b) \end{bmatrix} . \begin{bmatrix} \omega D \\ \omega E \end{bmatrix}$$

Desenvolvendo o produto central:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} (rD/2)cos\theta & (rE/2)cos\theta \\ (rD/2)sen\theta & (rE/2)sen\theta \\ (rD/b) & -(rE/b) \end{bmatrix} . \begin{bmatrix} \omega D \\ \omega E \end{bmatrix} \tag{B.6}$$

Discretizando (B.6) pelo método de euler:

$$\begin{bmatrix} X_{k+1} \\ Y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} X_k \\ Y_k \\ \theta_k \end{bmatrix} + T. \begin{bmatrix} (rD/2)cos\theta_k & (rE/2)cos\theta_k \\ (rD/2)sen\theta_k & (rE/2)sen\theta_k \\ (rD/b) & -(rE/b) \end{bmatrix} . \begin{bmatrix} \omega D_k \\ \omega E_k \end{bmatrix} \tag{B.7}$$

Onde T é o período de amostragem. Portanto:

$$\begin{aligned} X_{k+1} &= X_k + (rD.\omega D_k + rE.\omega E_k)T.cos\theta_k/2 \\ Y_{k+1} &= Y_k + (rD.\omega D_k + rE.\omega E_k)T.sen\theta_k/2 \\ \theta_{k+1} &= \theta_k + (rD.\omega D_k - rE.\omega E_k)T/2 \end{aligned}$$

As velocidades angulares das rodas podem ser medidas no período T contando o número de pulsos do *encoder* da roda direita ND e o número de pulsos da roda esquerda NE.

$$\begin{aligned} \omega D_k &= (2\pi/T).(C_g/C_e).ND_k \\ \omega E_k &= (2\pi/T).(C_g/C_e).NE_k \end{aligned}$$

Onde C_g é a redução mecânica de velocidade introduzida pelas engrenagens e C_e é a resolução do encoder em pulsos por revolução. Assim, o cinemático discretizado com odometria é dado por:

$$\begin{aligned} X_{k+1} &= X_k + (rD.ND_k + rE.NE_k).(\pi C_g/C_e).cos\theta_k \\ Y_{k+1} &= Y_k + (rD.ND_k + rE.NE_k).(\pi C_g/C_e).sen\theta_k \\ \theta_{k+1} &= \theta_k + (rD.ND_k - rE.NE_k).(2\pi C_g/C_e)/b \end{aligned} \tag{B.8}$$